

---

# **CHAINING SOFTWARE SECURITY VULNERABILITIES**

Martin Rakhmanov  
Security Research Manager  
Trustwave SpiderLabs

---

---

# WHO AM I?

- Security Research Manager  
@Trustwave SpiderLabs
- Reported various vulnerabilities in the past: databases (Microsoft SQL Server, Oracle Database, MySQL, SAP Adaptive Server Enterprise, IBM DB2 LUW), antiviruses (Avast, Sophos), routers (NETGEAR), hard drives (WD), service applications (Lenovo), mobile applications etc.



---

## WHY TALK ABOUT CHAINED VULNERABILITIES?

- Often developers consider “minor” vulnerabilities a small risk
  - Prove they are wrong: chaining several *small* bugs leads to a complete compromise!
-

---

## DATABASES: SAP ASE 16

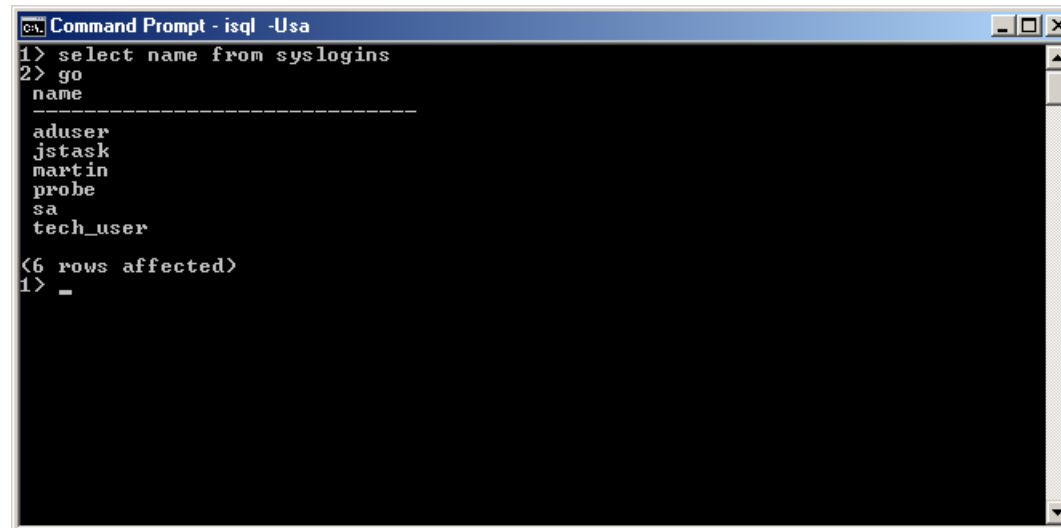


- A lot of built-in functionality: Java support, extended procedures, distributed transactions, Web Services etc.
  - Historically had many security issues, huge number reported by Trustwave (~36)
-

---

# VULNERABILITY I: "SPECIAL" LOGIN ACCESS

- `probe` account exists by default on any ASE and is enabled

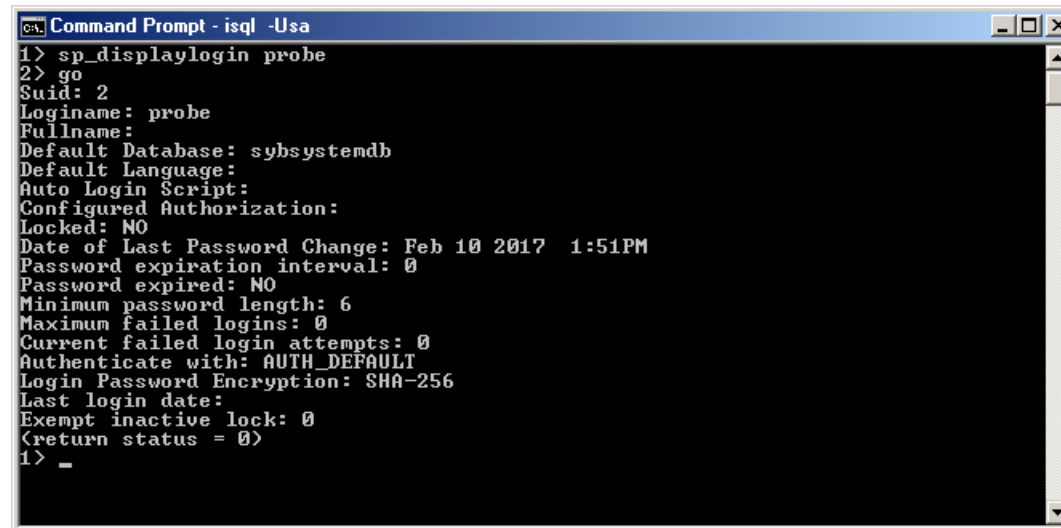


```
ca: Command Prompt - isql -Usa
1> select name from syslogins
2> go
name
-----
aduser
jstask
martin
probe
sa
tech_user
<6 rows affected>
1> _
```

---

# VULNERABILITY I CONTINUED

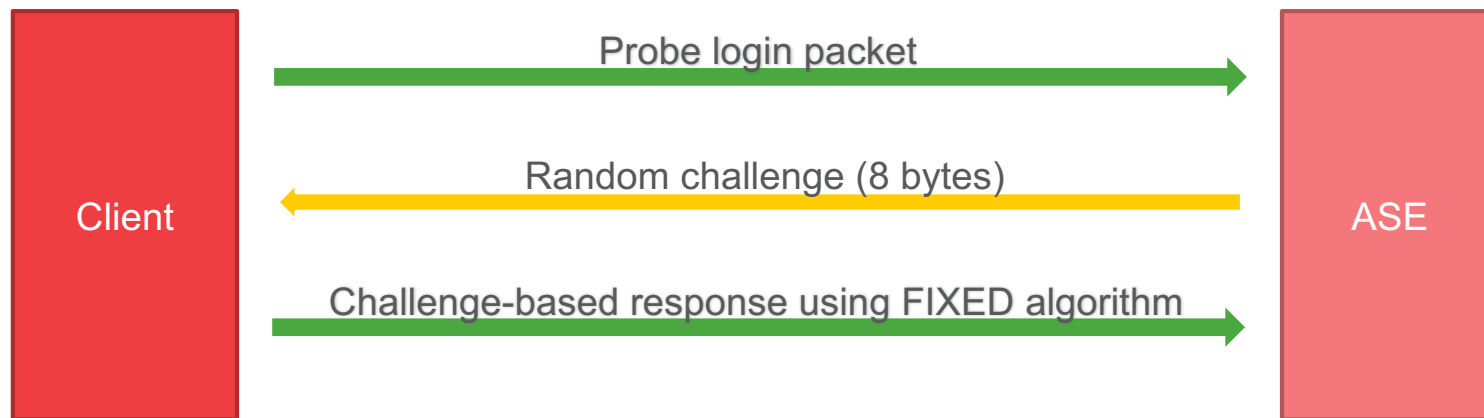
- This account uses special logon mechanism though...



```
CA: Command Prompt - isql -Usa
1> sp_displaylogin probe
2> go
Suid: 2
Loginame: probe
Fullname:
Default Database: sybsystemdb
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Feb 10 2017  1:51PM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts: 0
Authenticate with: AUTH_DEFAULT
Login Password Encryption: SHA-256
Last login date:
Exempt inactive lock: 0
(return status = 0)
1> _
```

---

# VULNERABILITY I CONTINUED



**What to do:** Never assume that some (secret) mechanism will not be uncovered

---

---

## VULNERABILITY II: FILESYSTEM ACCESS VIA JAVA

- There is a check that prevents write to locations under `$SYBASE`, but is that enough?
  - **What to do:** do not forget to implement all `SecurityManager` methods properly, think about **all** edge cases.
-



---

## VULNERABILITY III: CODE EXECUTION VIA JAVA

- **What to do:** again, implement all methods (`CheckLink`) of the `SecurityManager` class



---

# **CHAINED ATTACK ON SAP ASE**

1. Login as probe using provided library
  2. Write malicious payload to some location on server disk
  3. Load just written payload – executes as server process owner!
-

---

## ROUTERS: NETGEAR R8500

- Tri-Band WiFi Router with many functions: USB storage, UPnP, Cloud
- One attack surface is the Web Management facility.



---

# VULNERABILITY I: CSRF PROTECTION BYPASS

- Web management uses page-specific tokens to protect from CSRF
  - Exposed on some pages without authentication, aren't session-bound
  - `CSRF_TOKEN = SHA1(g_rand AND SHA1(page_name))`
  - Trivially brute-forced for all pages
  - **What to do:** do not expose sensitive data without authentication, bind CSRF to individual sessions.
-

---

# VULNERABILITY I: CSRF PROTECTION BYPASS

```
//-->
</SCRIPT>
<BODY bgColor=#ffffff leftMargin=0 topMargin=0 marginheight="0" marginwidth="0">
<form name="frmLanguage" method="POST" action="lang_top.cgi?id=fcd9975b3257ed36661f958d5f811e9609c57490">

<!--
  <input type="hidden" name="curlang" value="Auto">
-->

  <input type="hidden" name="curlang" value="English">
```

---

---

## VULNERABILITY II: THE *GENIEMAGIC*



- Authentication bypass by adding `genie` to the query string
  - Probably intended to allow the Genie app manage the device 😊
  - **What to do:** do NOT use hacks like “secret” strings in you apps!
-

---

## VULNERABILITY III: SHELL COMMAND INJECTION

- The `lan.cgi` page consumes some input without sanitizing
  - The `device_name` parameter can be abused to embed any shell command
  - Note that on the Web UI it has a limitation of 12 characters – but in reality it can be longer
  - **What to do:** sanitize all input and do NOT rely on client-side filters like HTML input length restrictions.
-

# VULNERABILITY III: SHELL COMMAND INJECTION

The screenshot displays the Netgear Genie web interface for a Nighthawk R8500 router. The 'ADVANCED' tab is selected, and the 'LAN Setup' page is active. A 'Device Name' input field is visible, containing the text 'R8500'. A Web Inspector overlay is positioned over the input field, showing the following HTML structure:

```
<form id="target" name="frmLan" method="POST" action="lan.cgi?id=a22b676ae130ca6ad97182b1dd5b3593d0f199e3">
  <input type="hidden" name="buttonHit">
  <input type="hidden" name="buttonValue">
  
  <div class="subhead2"> LAN Setup</div>
  <table border="0" style="height: 383px; position: relative; top: -3px; width: 928px;" class="subhead2-table">
    <tbody>
      <tr align="left" valign="middle">
        <tr>
          <td class="scrollpane-table-separate-border" colspan="2">
            <div class="scroll-pane" style="height: 383px; width: 928px; overflow: auto;">
              <table style="border-collapse: collapse; width: 97%">
                <tbody>
                  <tr>
                    <tr>
                      <td nowrap>
                        <td width="60%" align="right">
                          <input type="text" maxlength="12" size="24" name="device_name" value="R8500">
                        </td>

```



---

# CHAINED ATTACK ON NIGHTHAWK X8

1. Grab the `MNU_top.htm` page and extract current CSRF
  2. Using utility brute force `g_rand` counter value used to calculate CSRFs
  3. Calculate CSRF for the `lang.cgi` page
  4. Fetch the `lan.cgi` page passing CSRF along with genie and OS command
  5. Observe the router's filesystem for results
-

---

## Q&A

- <https://www.trustwave.com/Resources/SpiderLabs-Blog/>
  - <https://www.trustwave.com/Resources/Security-Advisories/>
  - Twitter: @mrakhmanov
-