

# Secure Programming Lecture 2: Landscape

David Aspinall

Informatics @ Edinburgh

# Course overview reminder

General organisation:

1. **Threats**
2. **Vulnerabilities**
3. **Defences**
4. **Processes**
5. **Methods**

We'll look at details under each of these, in various orders and combinations.

# Introduction

This lecture introduces the industry context behind software security vulnerability management.

- ▶ **threat** in a general setting
- ▶ **vulnerability** an example and its origin
- ▶ **defence** timeline of notifications, responses
- ▶ **process** security advisories and **CVE-IDs**
- ▶ **method** an organisational software security strategy

# Outline

An example vulnerability: overflow in X server

Vulnerabilities from the outside

Common Vulnerabilities and Exposures (CVEs)

Building Security In with BSIMM

Summary

# Threat

General aim: *services running on Unix systems should be robust against local and remote attackers.*

Otherwise: attackers may exploit a service to cause a DoS attack, gain access to a system, etc.

For a specific system, a threat analysis should consider the kinds of attackers and their motives (local? remote? what is being protected?) and then all the services running on the system.

**Question.** What's the easiest form of defence?

## A Vulnerability

A security review should first discover (and then monitor) relevant published **security advisories**.

For high value situations (and application code), dedicated review may be needed.

Jan. 7, 2014 - Stack buffer overflow in parsing of BDF font files in libXfont

**CVE-2013-6462:** An authenticated X client can cause an X server to read a font file that overflows a buffer on the stack in the X server, potentially leading to crash and/or privilege escalation in setuid servers. The fix is included in libXfont 1.4.7. See the advisory for more details.

# What is a BDF file?

```
STARTFONT 2.1
COMMENT
COMMENT Copyright (c) 1999, Thomas A. Fine
COMMENT
...
FONT -atari-small
SIZE 11 75 75
FONTBOUNDINGBOX 4 8 0 -1
STARTCHAR C000
ENCODING 0
SWIDTH 1 0
DWIDTH 4 0
BBX 4 8 0 -1
BITMAP
00
00
...
```

- ▶ BDF = **B**itmap **D**istribution **F**ormat
- ▶ A (mostly) obsolete font format by Adobe

## Advisory: **Description**

Scanning of the libXfont sources with the **cppcheck static analyzer** included a report:

```
[lib/libXfont/src/bitmap/bdfread.c:341]: (warning)
  scanf without field width limits can crash...
```

Evaluation of this report by X.Org developers concluded that a BDF font file containing a longer than expected string could **overflow the buffer on the stack**. Testing in X servers built with Stack Protector resulted in an immediate crash when reading a user-provided specially crafted font.

As libXfont is used to read user-specified font files in all X servers distributed by X.Org, including the Xorg server which is often run with root privileges or as setuid-root in order to access hardware, this bug may lead to an **unprivileged user acquiring root privileges** in some systems.



## Advisory: **Affected Versions**

This bug appears to have been introduced in the initial RCS version 1.1 **checked in on 1991/05/10, and is thus believed to be present in every X11 release starting with X11R5** up to the current libXfont 1.4.6. (Manual inspection shows it is present in the sources from the X11R5 tarballs, but not in those from the X11R4 tarballs.)

# The vulnerability in the code

```
338 char      charName[100];
339 int       ignore;
340
341 if (sscanf((char *) line, "STARTCHAR %s", charName) != 1) {
342     bdfError("bad character name in BDF file\n");
343     goto BAILOUT; /* bottom of function, free and return error */
344 }
```

# The vulnerability in the code

```
338 char      charName[100];
339 int       ignore;
340
341 if (sscanf((char *) line, "STARTCHAR %s", charName) != 1) {
342     bdfError("bad character name in BDF file\n");
343     goto BAILOUT; /* bottom of function, free and return error */
344 }
```

## SYNOPSIS

```
#include <stdio.h>
```

```
int sscanf(const char *str, const char
*format, ...);
```

## DESCRIPTION

`sscanf()` scans input from the character string pointed to by `str`, according to format string. This may contain conversions; results are stored in locations pointed to by the pointer arguments that follow `format`.

## Advisory: Fix

```
diff --git a/src/bitmap/bdfread.c b/src/bitmap/bdfread.c
index e2770dc..e11c5d2 100644
--- a/src/bitmap/bdfread.c
+++ b/src/bitmap/bdfread.c
@ -338,7 +338,7 @ bdfReadCharacters(FontFilePtr file, FontPtr pFont, b
    char        charName[100];
    int         ignore;

-   if (sscanf((char *) line, "STARTCHAR %s", charName) != 1) {
+   if (sscanf((char *) line, "STARTCHAR %99s", charName) != 1) {
        bdfError("bad character name in BDF file\n");
        goto BAILOUT; /* bottom of function, free and return error */
    }
```

The text above is an example of a *context diff* which shows the difference between two file versions. The **patch** command can be used to update the older file given this text. You need to know how to make and apply patches for this course. See 'man patch' on a Linux/Unix system.

# Defences

Options:

- ▶ Disable service
- ▶ Repair service: *downstream* updates
- ▶ Mitigate impact of attack

In running systems:

- ▶ Have there been past attacks?
- ▶ Can we check for future ones?

# Outline

An example vulnerability: overflow in X server

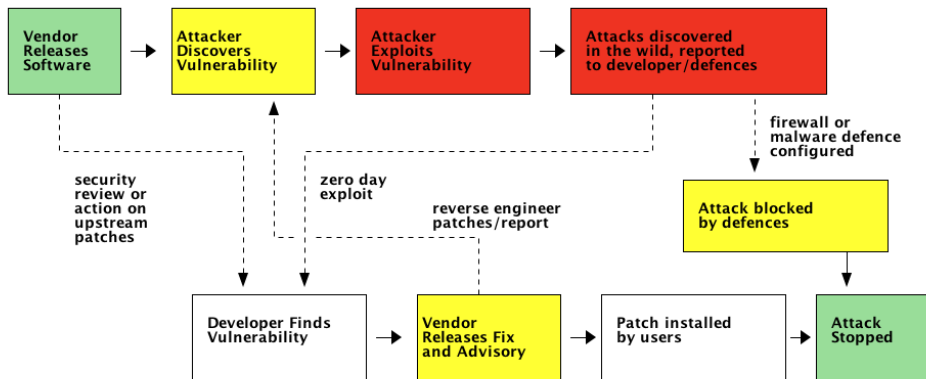
**Vulnerabilities from the outside**

Common Vulnerabilities and Exposures (CVEs)

Building Security In with BSIMM

Summary

# Vulnerability and attacks timeline



# Security advisories

Security **advisories** (aka **bulletins**) are issued by software vendors

- ▶ *public* feeds, also *private* at earlier stages
- ▶ advance notification to high-value customers, security companies
  - ▶ maybe before patches are available
  - ▶ (Q. is that a good idea?)
- ▶ public advisory usually when update available

Various people (sys admins, downstream software devs, users. . . ) should monitor and act on advisories.



# Security advisory format

Each vendor has own format. Typical information:

- ▶ Name, date, unique identification
- ▶ Criticality
- ▶ Affected products
- ▶ Solution

Varying amounts of information given:

- ▶ enough information to construct an exploit?
- ▶ if not, attackers may reverse engineer patches/updates anyway
- ▶ disclosure has to be planned carefully
  - ▶ typically by [coordinated disclosure](#)

## Advisory for libXfont vulnerability

Jan. 7, 2014 - Stack buffer overflow in parsing of BDF font files in libXfont

**CVE-2013-6462:** An authenticated X client can cause an X server to read a font file that overflows a buffer on the stack in the X server, potentially leading to crash and/or privilege escalation in setuid servers. The fix is included in libXfont 1.4.7. See the advisory for more details.

# Advisory on xorg-announce

X.Org Security Advisory: CVE-2013-6462: Stack buffer overflow in parsing of BDF font files in libXfont

Alan Coopersmith alan.coopersmith at oracle.com

Tue Jan 7 08:43:23 PST 2014

X.Org Security Advisory: January 7, 2014 - CVE-2013-6462

Stack buffer overflow in parsing of BDF font files in libXfont

=====

Description:

=====

Scanning of the libXfont sources with the cppcheck static analyzer included a report of:

```
[lib/libXfont/src/bitmap/bdfread.c:341]: (warning)
```

```
scanf without field width limits can crash with huge input data.
```

# Advisory on Red Hat enterprise-watch-list

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

=====  
Red Hat Security Advisory

Synopsis: Important: libXfont security update  
Advisory ID: RHSA-2014:0018-01  
Product: Red Hat Enterprise Linux  
Advisory URL: <https://rhn.redhat.com/errata/RHSA-2014-0018.html>  
Issue date: 2014-01-10  
CVE Names: CVE-2013-6462  
=====

## 1. Summary:

Updated libXfont packages that fix one security issue are now available for Red Hat Enterprise Linux 5 and 6.

The Red Hat Security Response Team has rated this update as having important security impact.

...

## 2. Relevant releases/architectures:

RHEL Desktop Workstation (v. 5 client) - i386, x86\_64

Red Hat Enterprise Linux (v. 5 server) - i386, ia64, ppc, s390x, x86\_64

Red Hat Enterprise Linux Desktop (v. 5 client) - i386, x86\_64

Red Hat Enterprise Linux Desktop (v. 6) - i386, x86\_64

Red Hat Enterprise Linux Desktop Optional (v. 6) - i386, x86\_64

Red Hat Enterprise Linux HPC Node (v. 6) - x86\_64

Red Hat Enterprise Linux HPC Node Optional (v. 6) - x86\_64

Red Hat Enterprise Linux Server (v. 6) - i386, ppc64, s390x, x86\_64

Red Hat Enterprise Linux Server Optional (v. 6) - i386, ppc64, s390x, x86\_64

Red Hat Enterprise Linux Workstation (v. 6) - i386, x86\_64

Red Hat Enterprise Linux Workstation Optional (v. 6) - i386, x86\_64

### 3. Description:

The libXfont packages provide the X.Org libXfont runtime library. X.Org is an open source implementation of the X Window System.

A stack-based buffer overflow flaw was found in the way the libXfont library parsed Glyph Bitmap Distribution Format (BDF) fonts. A malicious, local user could exploit this issue to potentially execute arbitrary code with the privileges of the X.Org server. (CVE-2013-6462)

Users of libXfont should upgrade to these updated packages, which contain a backported patch to resolve this issue. All running X.Org server instances must be restarted for the update to take effect.

#### 4. Solution:

Before applying this update, make sure all previously-released errata relevant to your system have been applied.

This update is available via the Red Hat Network. Details on how to use the Red Hat Network to apply this update are available at <https://access.redhat.com/kb/docs/DOC-11259>

#### 5. Bugs fixed (<https://bugzilla.redhat.com/>):

1048044 - CVE-2013-6462 libXfont: stack-based buffer overflow flaw when parsing Glyph Bitmap Distribution Format (BDF) fonts

## 6. Package List:

Red Hat Enterprise Linux Desktop (v. 5 client):

Source:

<ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Client/en/os/SRPMS/>

i386:

libXfont-1.2.2-1.0.5.el5\_10.i386.rpm

libXfont-debuginfo-1.2.2-1.0.5.el5\_10.i386.rpm

...

...



## 7. References:

<https://www.redhat.com/security/data/cve/CVE-2013-6462.html>  
<https://access.redhat.com/security/updates/classification/#important>

## 8. Contact:

The Red Hat security contact is <secalert redhat com>. More contact details at  
<https://access.redhat.com/security/team/contact/>

Copyright 2014 Red Hat, Inc.

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.4 (GNU/Linux)

iD8DBQFSz8HSXlSAg2UNWIIRAv05AJ4976ATNgp8mmoyRg0bDFnCv0P4zACfYWJc  
f9VhkwpGzE3y3jtSD9fupVg=  
=T7Wm

-----END PGP SIGNATURE-----

# Example: HP Data Protector

## HP Support document

SEARCH HP SUPPORT CENTER ▶

Help

### RELATED LINKS

HP Software Support Online (IT Management Software)

HP Customer Care (Home & Home Office products)

Subscribe to alerts for your product

[Rate this content](#)

### **SUPPORT COMMUNICATION - SECURITY BULLETIN**

**Document ID:** c03822422

**Version:** 1

**HPSBMU02895 SSRT101253 rev.1 - HP Data Protector, Remote Increase of Privilege, Denial of Service (DoS), Execution of Arbitrary Code**

**NOTICE:** The information in this Security Bulletin should be acted upon as soon as possible.

**Release Date:** 2014-01-02

**Last Updated:** 2014-01-02

**Potential Security Impact:** Remote increase of privilege, Denial of Service (DoS), execution of arbitrary code

**Source:** Hewlett-Packard Company, HP Software Security Response Team

### **VULNERABILITY SUMMARY**

Potential security vulnerabilities have been identified with HP Data Protector. These vulnerabilities could be remotely exploited to allow an increase of privilege, create a Denial of Service (DoS), or execute arbitrary code.

### **References:**

- CVE-2013-2344 (ZDI-CAN-1866, SSRT101217)
- CVE-2013-2345 (ZDI-CAN-1869, SSRT101218)
- CVE-2013-2346 (ZDI-CAN-1870, SSRT101219)

# What is HP Data Protector?

**Big data causes big backup challenges**



# How was this vulnerability found?



- ▶ **Zero Day Initiative**, started by TippingPoint, a network security company
  - ▶ part of 3Com, now HP
- ▶ Idea of *crowd-sourcing* vulnerability discovery
- ▶ Finding many vulnerabilities in enterprise software
  - ▶ HP, Microsoft, CISCO, . . .
- ▶ Incentive programme rewarding participants
  - ▶ \$ reward, bonuses like DEFCON attendance
  - ▶ advantages: independence, wider knowledge
  - ▶ and presumably cheaper than direct employment

# Outline

An example vulnerability: overflow in X server

Vulnerabilities from the outside

**Common Vulnerabilities and Exposures (CVEs)**

Building Security In with BSIMM

Summary

# What is CVE?

- ▶ Started in 1999, originally at [CERT](#)
  - ▶ CVE = Common Vulnerability *Enumeration*
- ▶ Aim: standardise identification of vulnerabilities
  - ▶ vendor's own schemes: confusion, duplication
- ▶ Each vendor/distributor has own advisory channel
  - ▶ CVE allows cross referencing, public standard ID
  - ▶ Users or customers can check how CVEs are handled
- ▶ CVEs handled by [MITRE](#), a US R& D outfit
  - ▶ CVE = **Common Vulnerabilities and Exposures**
- ▶ US [National Vulnerability Database, NVD](#) at NIST
  - ▶ CVEs feed the NVD
- ▶ [ITU-T 2011](#): X.CVE international recommendation

# Vulnerabilities versus Exposures

**Vulnerability** A mistake that can be used by a hacker to violate a “reasonable” security policy for a system (e.g., executing commands as another user, violating access restrictions, conducting a DoS attack)

Example: smurf vulnerability (ping server responds to broadcast address)

**Exposure** A system configuration issue or mistake in software that can be used by a hacker as a stepping-stone into a system or network, e.g., gathering information, hiding activities.

Example: running open ‘finger’ service; allows attacker to probe network

# CVE Identifiers

Consist of:

- ▶ CVE ID (number): **CVE-1999-0067**
- ▶ Brief description of vulnerability or exposure
- ▶ References, e.g., to reports or advisories



# Creating CVE Identifiers

1. Discover a potential V or E
2. Get a CVE Numbering Authority to give a number
  - ▶ MITRE, big vendors (Apple, Google, MS, Ubuntu, . . .)
  - ▶ Numbers reserved in blocks; “instantly” available
3. CVE ID number shared among disclosure parties
4. Advisory published, including CVE-ID number
5. MITRE updates master list

Only published CVE-ID Numbers are kept in master list.

# CVE Compatibility

- ▶ Standard for “interoperability” or “comparability”
- ▶ For products and services
- ▶ Has some official requirements certified by MITRE
  - ▶ ownership by legal entity
  - ▶ responsibility, answering to reviews
- ▶ Capability required for tools, web sites
  - ▶ *CVE searchable*
  - ▶ Use standard document formats

# Outline

An example vulnerability: overflow in X server

Vulnerabilities from the outside

Common Vulnerabilities and Exposures (CVEs)

**Building Security In with BSIMM**

Summary

# BSIMM: Building Security In Maturity Model



# BSIMM: Building Security In Maturity Model

- ▶ **BSIMM** is a *Maturity Model* for real-world best practices in software-producing companies
  - ▶ examines *Software Security Initiatives* (SSIs)
  - ▶ data-driven: defined by survey results
  - ▶ provides a “measuring stick”, state-of-the-art
- ▶ Introduced by Gary McGraw and others
  - ▶ Author of *Software Security: Building Security In*
- ▶ Now at **BSIMM-14**, December 2023. Has 126 software sec activities.
- ▶ Inspired by **Capability Maturity Model (CMM)** (late 80s-90s)
  - ▶ model of software development processes
  - ▶ *maturity* = degree of formality/rigour of process
  - ▶ 5 Levels: *chaotic, repeatable, defined, managed, optimizing*

## BSIMM goals

For organisations starting/running a Software Security Initiative, BSIMM aims to:

- ▶ Inform risk management decisions
- ▶ Clarify “right thing to do” for those involved
- ▶ Reduce costs via standard, repeatable processes
- ▶ Improve code quality

This is done by planning a *Software Security Initiative*, implementing activities selected from BSIMM. Activities can be roled out according to the maturity level of the organisation.

# Implementing a SSI

A Software Security Initiative may be a **serious effort** for a large organisation to implement, and require a big budget.

Large companies can have:

- ▶ tens of thousands of software developers
- ▶ hundreds or thousands of applications in development
- ▶ similarly many applications in deployment or sale

Systematic, explicit organisation of security goals are needed to manage software security effectively.

# BSIMM Software Security Framework

BSIMM defines a *Software Security Framework* to describe SSIs

- ▶ **4 domains** covering **12 practices**
- ▶ Each practice involves numerous **activities**
- ▶ Activities are assigned **maturity levels 1-3**
  - ▶ 1: most mature, common everywhere
  - ▶ 3: least mature, emerging (or expiring) activity
- ▶ Covers **126** activities
- ▶ New activities added when they appear in >1 org



# BSIMM domains



GOVERNANCE



INTELLIGENCE



GOVERNANCE



INTELLIGENCE

## BSIMM evolves over time

Results are used to identify most common (core) activities in each practice and discover new activities, as well as adjust levels and possibly remove activities that become unpopular.

Example updates:

**BSIMM 6** *CMVM3.4 Operate a bug bounty programme* added.

**BSIMM 7** *SE3.4 Use application containers*

**BSIMM 9** *SE3.7 Ensure cloud security basics* added.  
Secure coding standards increases to level 3.

**BSIMM 12:** *SE3.8 App composition analysis on code repos* added.

**Exercise.** Suggest reasons for each of these changes.

# Outline

An example vulnerability: overflow in X server

Vulnerabilities from the outside

Common Vulnerabilities and Exposures (CVEs)

Building Security In with BSIMM

**Summary**

# Review questions

## Server vulnerabilities

- ▶ Explain why services running on Unix systems should be robust against both local and remote attackers, even if local users are trusted.

## Patches, updates, defences

- ▶ Explain the lifecycle of a software vulnerability. Consider cases where the vulnerability is found by a “black-hat” or by a “white-hat” first.

## Software security processes

- ▶ Explain *vendor security advisories* and CVEs.
- ▶ Discuss the role of **BSIMM** in improving software security development practices in industry and give example activities in each of its 12 practices.