Основы безопасности информационных технологий

Мир уязвимостей

Содержание лекции

- Тенденции мира уязвимостей
- Переполнение буфера
- Внедрение команд
- SQL инъекции

Q3 2010 Vulnerability Research Tracker

- число зафиксированных публично уязвимостей возросло на 62% по сравнению с 2009-м годом
- уязвимостей высокой степени риска было зафиксировано большинство - почти 70%
- Adobe, MS Office, RealPlayer, MS IE и Apple Safari.
- бизнес- и мультимедиа-приложения
- Windows, MacOS, Linux
- наиболее часто встречаются уязвимостей, связанные с буферами. На втором месте code injection.
- результатом использования большинства (около 50%) уязвимостей является выполнение некоего кода. На втором месте - отказ в обслуживании.

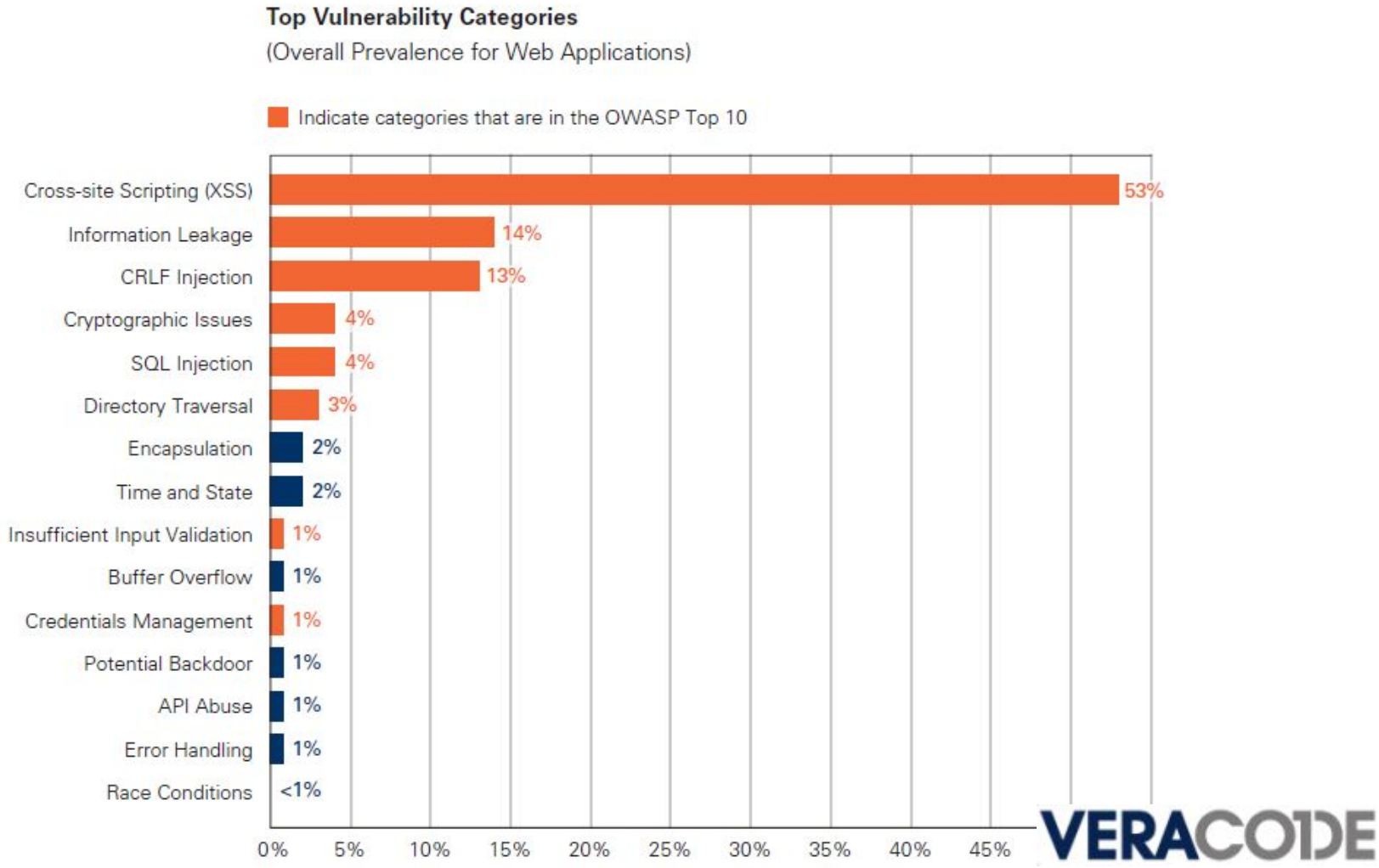
FROST & SULLIVAN

The State of Software Security Report

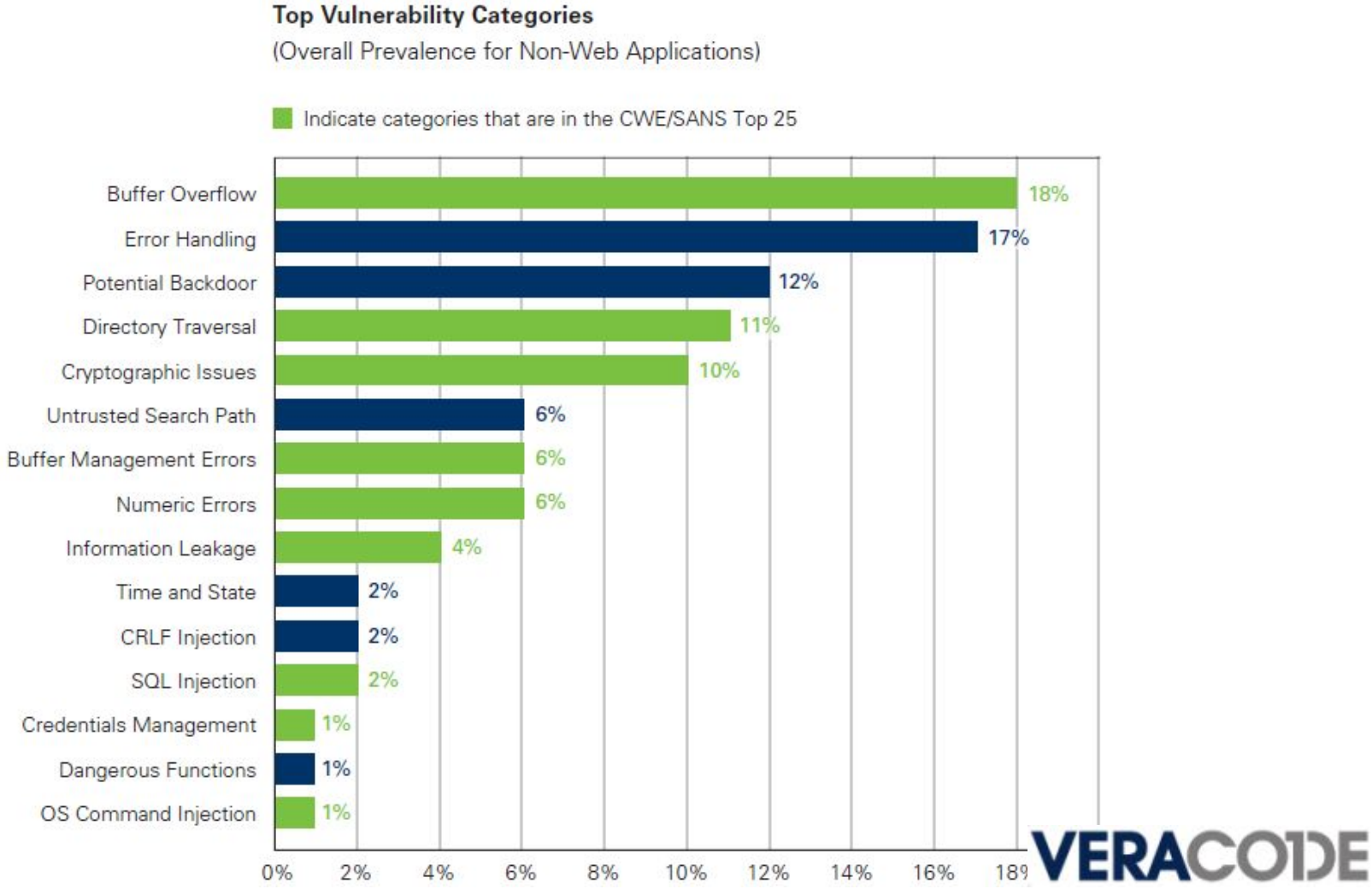
- 8 из 10 Web-приложений провалили "тест" OWASP Top 10 (The Open Web Application Security Project)
- более половины всех приложений имеют практически нудевой уровень качества защиты ПО
- CSS остается одной из распространенных проблем
- большинство разработчиков остро нуждается в тренингах по вопросам ИБ
- лучше всех защищены финансовые организации
- даже разработчики средств защиты не следуют правилам безопасного программирования
- для анализа ПО необходимо использовать и статический и динамический анализ

VERACODE

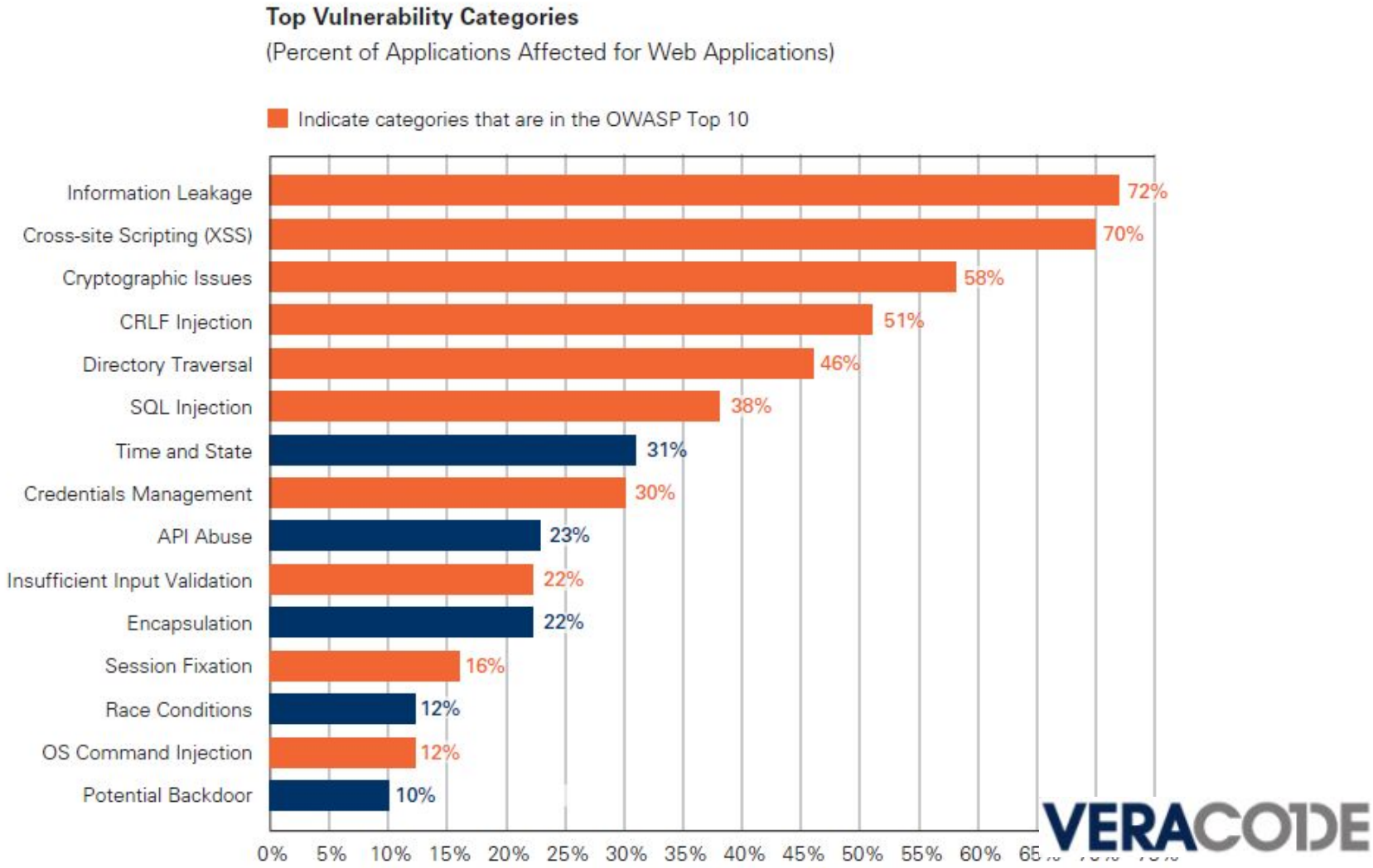
The State of Software Security Report



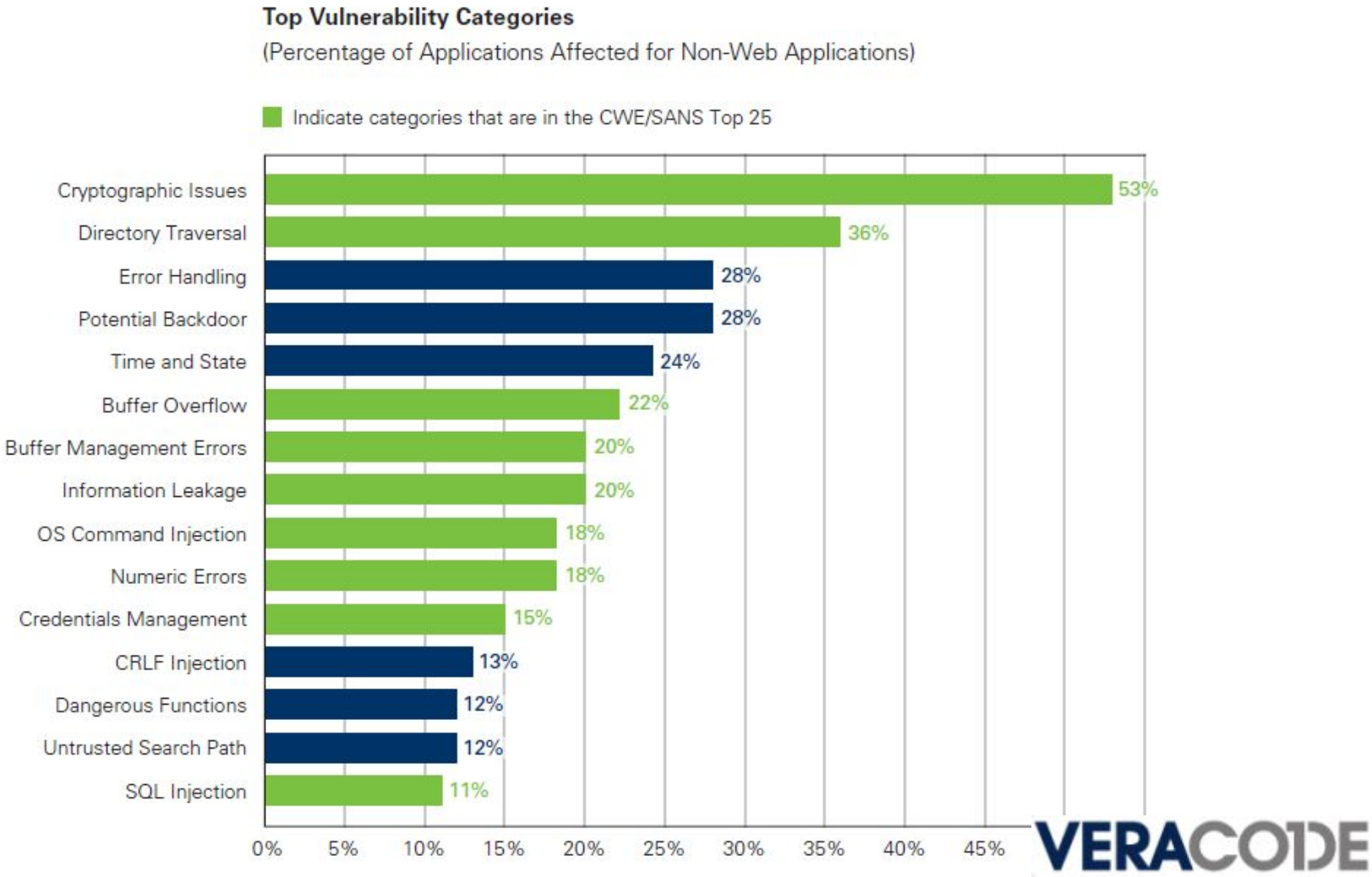
The State of Software Security Report



The State of Software Security Report



The State of Software Security Report



The State of Software Security Report

Vulnerability Distribution by Language

Java		ColdFusion		C/C++		.NET		PHP	
Cross-site Scripting (XSS)	50%	Cross-site Scripting (XSS)	89%	Buffer Overflow	27%	Cross-site Scripting (XSS)	44%	Cross-site Scripting (XSS)	80%
CRLF Injection	17%	SQL Injection	9%	Error Handling	23%	Information Leakage	23%	Directory Traversal	8%
Information Leakage	14%	OS Command Injection	<1%	Potential Backdoor	22%	Cryptographic Issues	11%	SQL Injection	6%
Cryptographic Issues	5%	Information Leakage	<1%	Numeric Orders	11%	Directory Traversal	8%	Information Leakage	3%
Directory Traversal	4%	Directory Traversal	<1%	Buffer Mgmt Errors	9%	Insufficient Input Validation	6%	Code Injection	1%

VERACODE

Переполнение буфера

```
#include <stdio.h>

void DontDoThis(char* input) {
    char buf[16];
    strcpy(buf, input);
    printf("%s\n", buf);
}

int main(int argc, char* argv[]) {
    DontDoThis(argv[1]);
    return 0;
}
```

Переполнение буфера

0x0012FEC0 c8 fe 12 00 D 0.. <- адрес аргумента buf

0x0012FEC4 c4 18 32 00 D .2. <- адрес аргумента input

0x0012FEC8 d0 fe 12 00 D D.. <- начало буфера buf

0x0012FECC 04 80 40 00 .«Unicode: 80»@.

0x0012FED0 el 02 3f 4f D .?0

0x0012FED4 66 00 00 00 f... <- конец buf

0x0012FED8 e4 fe 12 00 0 0.. <- содержимое регистра EBP

0x0012FEDC 3f 10 40 00 ?.@. <- адрес возврата

0x0012FEE0 c4 18 32 00 0.2. <- адрес аргумента DontDoThis

0x0012FEE4 c0 ff 12.00 0 0..

0x0012FEE8 10 13 40 00 . .@. <- адрес, куда вернется main()

Переполнение буфера

Важно обращать внимание на:

- любые входные данные
- передачу входных данных во внутренние структуры
- использование небезопасных функций работы со строками
- использование арифметических операций для вычисления размера буфера

Переполнение буфера

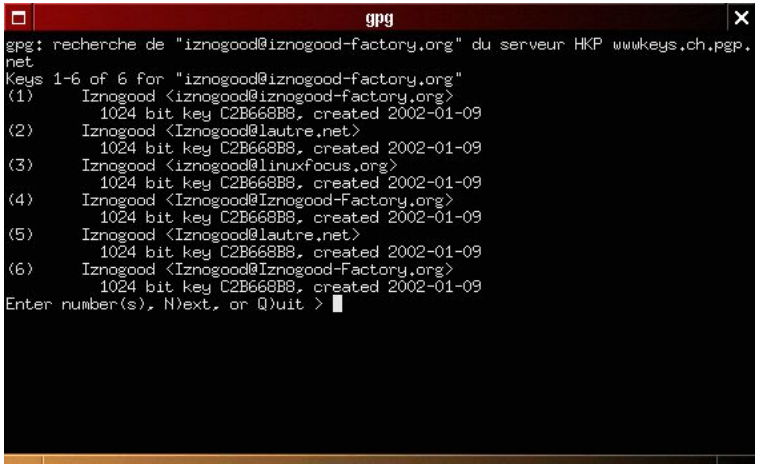
Как избежать переполнение?

- Замена опасных функций работы со строками
- Контроль за выделением памяти
- Проверка циклов и обращений к массивам
- Замена строковых буферов C строками C++
- Замена статических массивов контейнерами STL (Standard Template Library)
- Использование инструментов анализа

Внедрение команд

```
char buf[1024];
snprintf(buf, "system lpr -P %s", user_input, sizeof(buf) - 1);
system(buf);
```

FRED; xterm&



Внедрение команд

```
def call_func(user_input, system_data):
    exec 'special_function_%s("%s")' % (system_data, user_input)

system_data = sample, a user_input = fred

special_function_sample(«fred»)

user_input = fred"); print("foo

special_function_sample("fred"); print("foo")
```

SQL инъекции

MySQL

PostgreSQL

ORACLE

SYBASE

Microsoft SQL Server 2008

SQL инъекции

```
using System.Data;
using System.Data.SqlClient;
string ccnum = "None";
try {
    SqlConnection sql = new SqlConnection(
        @"data source=localhost;" +
        "user id=sa;password=pAs$wOrd;");
    sql.Open();
    string sqlstring="SELECT ccnum" +
    " FROM cust WHERE id=" + Id;
    SqlCommand cmd = new SqlCommand(sqlstring,sql);
    ccnum = (string)cmd.ExecuteScalar();
} catch (SqlException se) {
    Status = sqlstring + " failed\n\r";
    foreach (SqlError e in se.Errors)
        Status += e.Message + "\n\r";}
}
```

SQL инъекции

```
using System.Data;
using System.Data.SqlClient;
string ccnum = "None";
try {
    SqlConnection sql = new SqlConnection(
        @"data source=localhost;" +
        "user id=sa;password=pAs$wOrd;");
    sql.Open();
    string sqlstring="SELECT ccnum" +
        " FROM cust WHERE id=%ID%";
    String sqlstring2 = sqlstring.Replace("%ID", id);
    SqlCommand cmd = new SqlCommand(sqlstring2,sql);
} catch (SqlException se) {
    Status = sqlstring + " failed\n\r";
    foreach (SqlError e in se.Errors)
        Status += e.Message + "\n\r";}
}
```

SQL инъекции

```php
<?php
    $db = mysql_connect("localhost","root","$$sshhh...!");
    mysql_select_db("Shipping",$db);
    $id = $HTTP_GET_VARS["id"];
    $qry = "SELECT ccnum FROM cust WHERE id = %$id%";
    $result = mysql_query($qry,$db);
    if ($result) {
        echo mysql_result($result,0,"ccnum");
    } else {
        echo "No result! " . mysql_error();
    }
?>
```

SQL инъекции

```
#!/usr/bin/perl

use DBI;
use CGI;

print CGI::header();
$cgi = new CGI;
$id = $cgi->param('id');

print "<html><body>";
$dbh = DBI->connect('DBI:mysql:Shipping:localhost', 'root', '$3cre+')
or print "Ошибка connect : $DBI::errstr";
$sql = "SELECT ccnum FROM cust WHERE id = " . $id;
$sth = $dbh->prepare($sql)
or print "Ошибка prepare : $DBI::errstr";
$sth->execute ()
or print "Ошибка execute : $DBI::errstr";

while (@row = $sth->fetchrow_array ) {
    print "@row<br>";
}

$dbh->disconnect;
print "</body></html>";
exit:
```

SQL инъекции

Сопровождающие ошибки:

- подключение с привилегированной учетной записью;
- встраивание пароля в программный код;
- сообщение противнику излишне подробной информации в случае ошибки.

SQL инъекции

Признаки уязвимости:

- приложение получает данные от пользователя;
- приложение не проверяет корректность входных данных;
- приложение использует введенные пользователем данные для запроса к базе;
- приложение применяет конкатенацию или замену подстроки для построения SQL-запроса;
- приложение пользуется командой SQL exec (или ей подобной).