E-BANKING SYSTEMS

VULNERABILITY STATISTICS FOR E-BANKING SYSTEMS (2011—2012) WHITE PAPER

CONTENTS

EXECUTIVE SUMMARY

This report provides a summary of research carried out by Positive Research, the research division of Positive Technologies, into vulnerabilities in e-banking systems. For the purposes of this regional study, only e-banking systems used in Russia were included. Other regions use different systems and may have varying results.

To learn more, please contact Positive Technologies.

KEY FINDINGS:

1. One in three e-banking systems has only low to medium security levels.

Our specialists were able to access the OS or DBMS of one out of every three e-banking systems analyzed. Making it possible to take full control of the system – allowing the execution of unauthorized online transactions, the potential for denial of service and even the likelihood of withdrawing monies, from accounts that are not setup for e-banking. In fact, we successfully executed unauthorized transactions on 37% of e-banking systems, in the study. A few medium-risk vulnerabilities are all that an intruder might need to execute fraudulent transactions; which we found to be present in all the systems under examination.

2. Half of the systems tested were at high risk.

Of all the vulnerabilities detected, 8% were high severity vulnerabilities, 51% were classed as medium severity vulnerabilities, and 41% were considered to be of low severity. Whilst this means low and medium severity vulnerabilities are by far the most common, the 8% of high-risk vulnerabilities found were spread across half of the systems tested. This means a staggering 50% of the e-banking systems we analyzed were at high risk of attack.

The most common vulnerabilities are weak password policy and weak protection against brute force attacks. A remarkable 82% of the assessed systems were exposed to this kind of weakness. Although these are not considered high risk vulnerabilities when taken individually, a malware user can often carry out unauthorized transactions on the user level by exploiting several medium severity vulnerabilities. We must therefore draw the conclusion that an absence of high severity vulnerabilities does not mean that an e-banking system is well protected.

3. Off-the-shelf applications contain 4X more vulnerabilities than in-house ones.

Commercially available e-banking applications were found to contain up to four times as many vulnerabilities than that of their home-grown counterparts. And high-risk web application vulnerabilities were only found in those systems that utilized off-the-shelf solutions. Such applications are usually cross-platform, employ a complex architecture and have many functions; making them difficult to be written securely

4. E-banking Application Security Degrades After Launch.

According to our research, systems in-use actually accumulate more weaknesses, over time. We found that production systems tested reported up to 1.5 times

more vulnerabilities, at all severity levels, than their pre-production counterparts. These new vulnerabilities can arise from incorrect configurations, inadequate authentication methods and errors within the application source code. Our findings illustrate that it is not enough to simply check the security of an e-banking system when it is first launched, but regular audits are also needed once in use.

5. 100% of the e-banking systems demonstrated flawed security mechanisms. More than 60% of the e-banking systems included in this study were found to contain at least one user identification flaw – either a predictable UID format or disclosure of information about system UIDs. All the systems had authentication flaws including weak password policies or inadequate protections against brute force attacks. Moreover, 80% of the systems exhibited at least one problem with user authorization, with some lacking multi-factor authentication when completing a transaction. And while not requiring another factor itself does not independently threaten the system, it makes it much easier for an intruder to use a combination of weaknesses to gain unauthorized access to accounts.

1. SOURCE DATA & METHODOLOGY

Our research covers 11 e-banking systems which were analyzed by the experts at Positive Research in 2011 and 2012. This report includes only our findings on those systems for which we conducted comprehensive analysis, including tests on operating logic as well as the source code itself. It therefore does not include the results of tests we carried out on systems which were still under development as these were only assessed for application code vulnerabilities, without any analysis of operational weaknesses that might allow unauthorized transactions to be carried out.

The e-banking systems analyzed for this report can be divided into two types: those offering services to private individuals and those serving companies. More than 70% of them serve individuals. Only three of them required the user to download any component of the e-banking application onto their own device. The remainder allowed the user to connect and carry out transactions through a standard web browser.

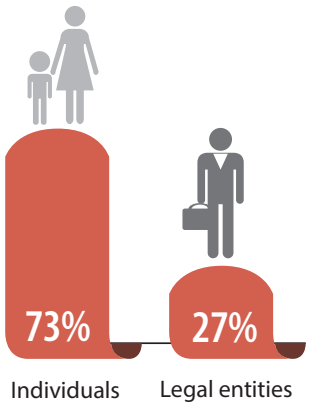

Figure 1. E-banking systems distribution by service areas

55% of the systems assessed were based on commercially available off-the-shelf solutions. While less than half were solutions developed in-house, by individual banks: three systems used Java and two were written in C# and PHP.

Figure 2 illustrates the division of systems by provider and programming languages.

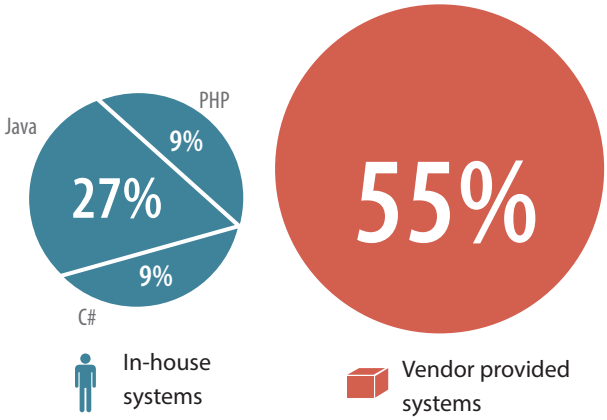

Figure 2. Off-the-shelf vs. in-house development and programming language

Critical vulnerabilities were found among the systems purchased by banks from well-known vendors. In accordance with our responsible disclosure policy, this report does not specify the names of such vendors.

All the systems assessed in the course of this research were different software products. Moreover, the results of a second security analysis of e-banking systems - which was carried out mostly to check whether the vulnerabilities we had detected had since been eliminated – were not included in the statistics. This was to avoid any duplication of results which would have rendered the study less useful as an objective estimation of current e-banking system security in Russia. Given the popularity of the vendor provided banking systems with other financial institutions, the results of this research have implications for potentially hundreds of banks worldwide.

The research detailed in this report included analysis of production systems, pre-production and test environments. More than a half of the systems were examined in pre-production environments – either those in the final stages of testing prior to launch, or fully functioning copies of production systems that were already in use. One of the systems was examined both on the test platform and during operation. Figure 3 illustrates the division of systems by platform type.

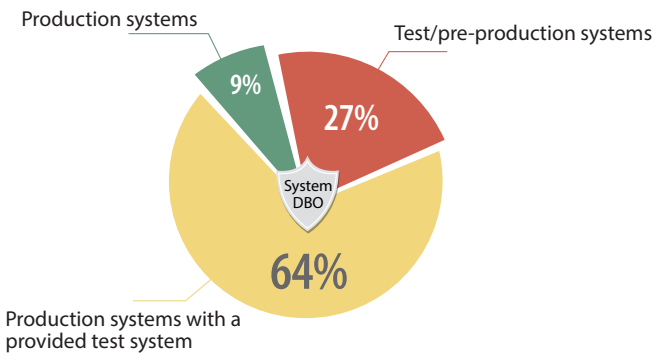

Figure 3. Tested systems by platform type

All but one of the production systems provided for analysis were solutions provided by commercial vendors of e-banking systems. In-house systems were mostly analyzed in pre-production before they were put into operation; while their commercial counterparts were most often not checked until after they were placed into production.

2. OVERALL RESULTS

2.1 The Most Common Vulnerabilities and Related Threats

Our security analysis of e-banking revealed many vulnerabilities across varying severity levels. In total, 8% of all detected vulnerabilities were identified as high-severity (see figure 4). The largest percentage of the vulnerabilities found (51%) were of medium severity and a significant number of vulnerabilities (41%) were classified as low severity.

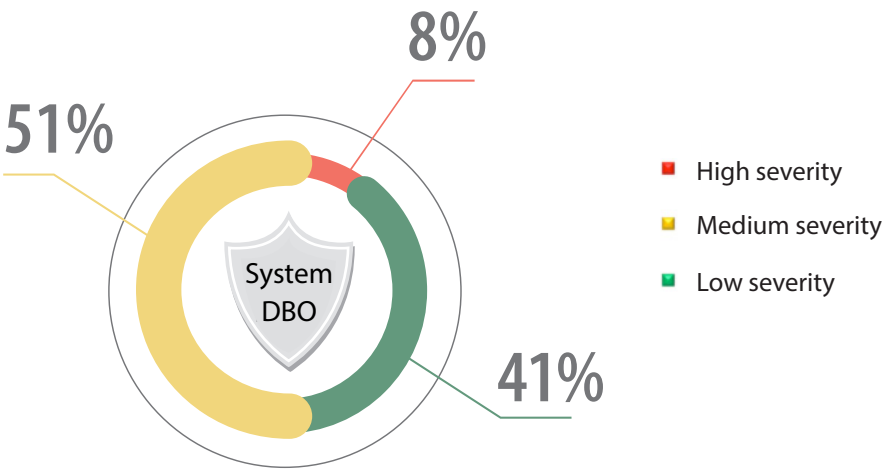

Figure 4. Vulnerabilities distribution by severity

In the majority of cases, the vulnerabilities detected were connected with flaws in security mechanisms such as the process for identifying, authenticating and authorizing users (43%), as well as with systems configuration flaws (34%) and application source code vulnerabilities (23%). Figure 5 shows the percentage of various vulnerability types.

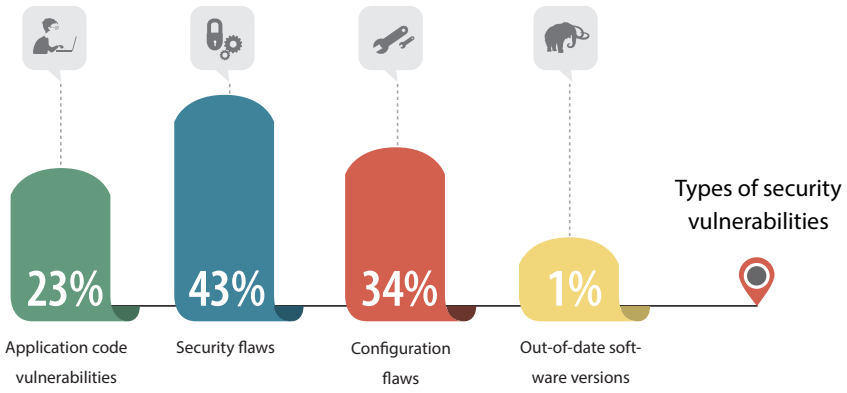

Figure 5. Vulnerabilities found, classified by vulnerability type

The most common vulnerabilities were connected with weak password policies (82%) and inadequate protections from brute force attacks (82%). Disclosure of information about the implemented software was also common in many of the systems (73%) – availability of such data makes life much easier for hackers. When it comes to application source code vulnerabilities, flaws resulting in cross-site scripting were widely observed (62%). This type of weakness facilitates attacks against users (for instance, using social engineering techniques). Figure 6 provides details of the ten vulnerabilities most commonly found in the e-banking systems studied.

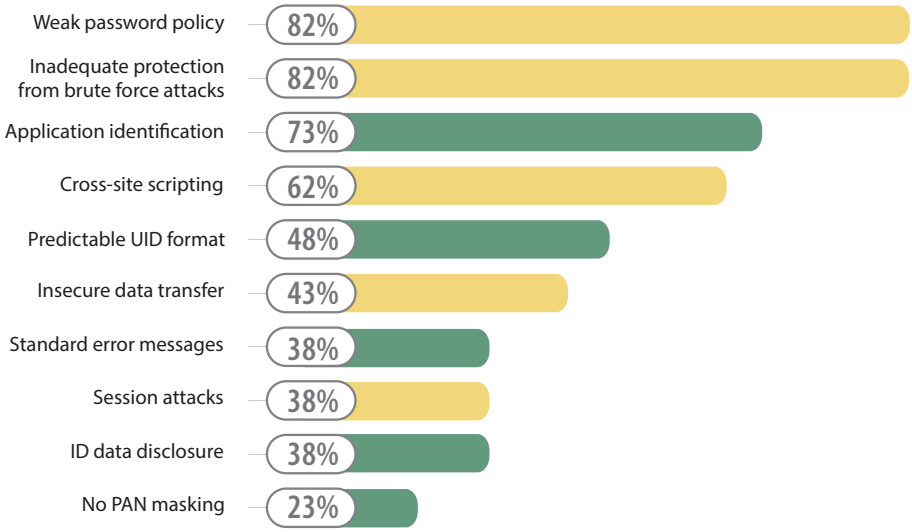

Figure 6. The most commonly-found vulnerabilities in the e-banking systems analyzed

While according to the research, the majority of the vulnerabilities found were of medium and low severity, if exploited together they could have the same grave consequences as one of the less common high-severity flaws. They may even allow hackers to obtain full control over the e-banking system. Section 3 of this report (page 14) provides descriptions and information on the vulnerabilities with the highest risk levels.

Figure 7 illustrates the types of access a hacker could obtain by exploiting the vulnerabilities we found during the course of our research.

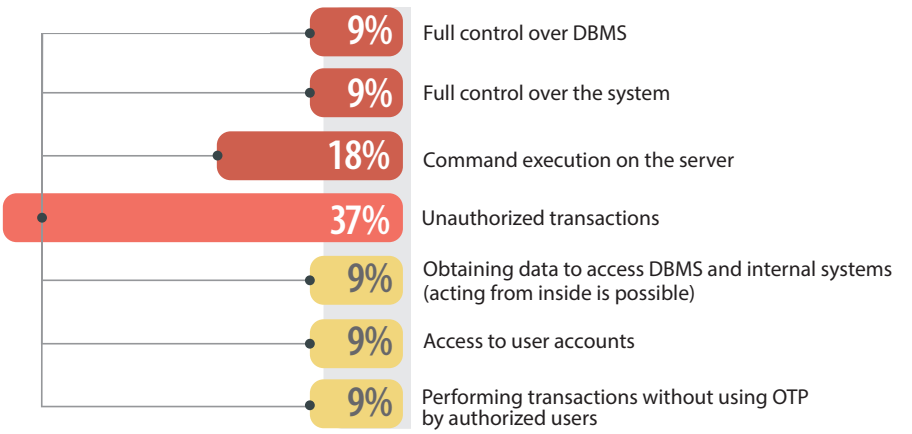

Figure 7. Possible consequences of an attack exploiting the vulnerabilities found during this research

In more than 70% of cases, an attacker could either access an operating system or DBMS, at the server level, or carry out unauthorized user transactions by exploiting one of the vulnerabilities we detected. Both in-house systems and commercially available ones had vulnerabilities which could result in these types of attack. An attacker using malware needs only to exploit a few medium-severity vulnerabilities together to carry out unauthorized user transactions. Therefore, the mere absence of high-severity vulnerabilities does not assure a system is well protected.

According to figures provided by the financial institutions involved in this research, the number of people that use the e-banking systems we tested ranges from tens of thousands to hundreds of thousands of users per system and can be seen in figure 8 below. For systems analyzed on test platforms, we have specified the number of users after the system was brought to operation.

We can therefore conclude that if the e-banking vulnerabilities we detected were exploited by attackers, many thousands of users could become victims of cyber fraud, leaving the banks involved facing considerable financial penalties as well as significant damage to their reputation.

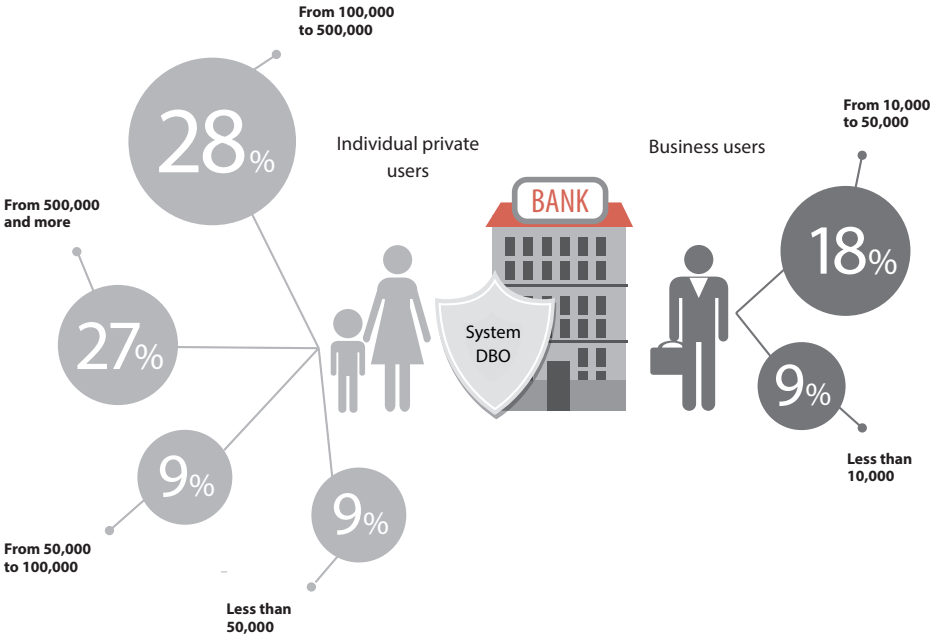

Figure 8. The number of e-banking system users

2.2 E-banking System Vulnerabilities for Individuals and Organizations

Application source code vulnerabilities are more common in systems used by businesses versus those used by individual consumers (38% vs. 18%). Although, systems used by individual consumers contained more vulnerabilities connected with configuration flaws (37% vs. 24%), than their business counterparts. Figure 9 provides a breakdown of vulnerability types based on this market segmentation.

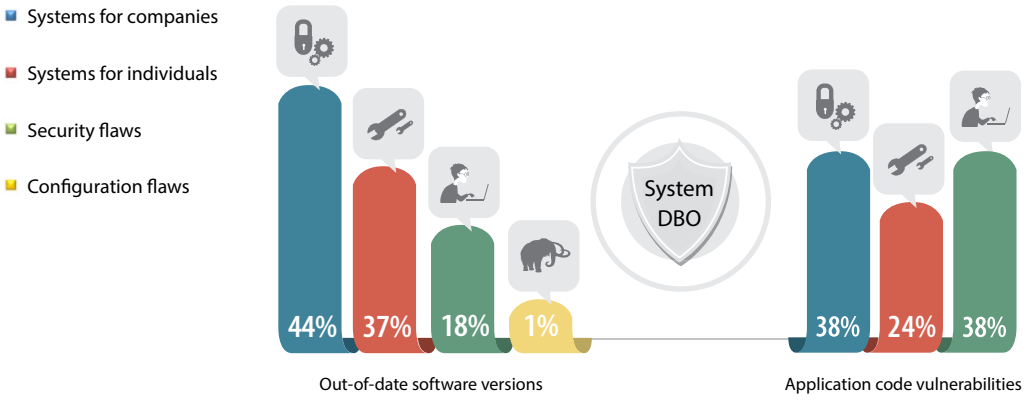

Figure 9. Vulnerability statistics for systems aimed at individuals and organizations

The mix of vulnerability severity levels is virtually the same between systems for individual consumers and those for businesses (see figures 10 and 11).

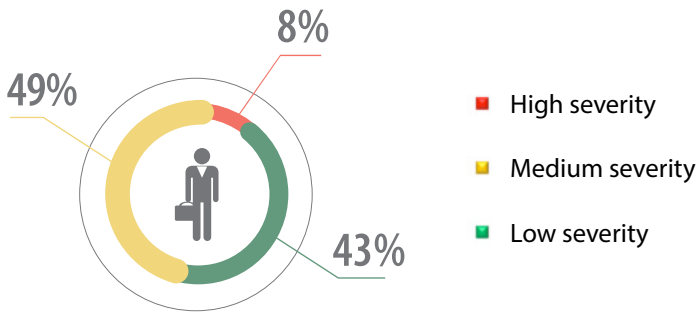

Figure 10. Vulnerability distribution by severity (for systems aimed at businesses)

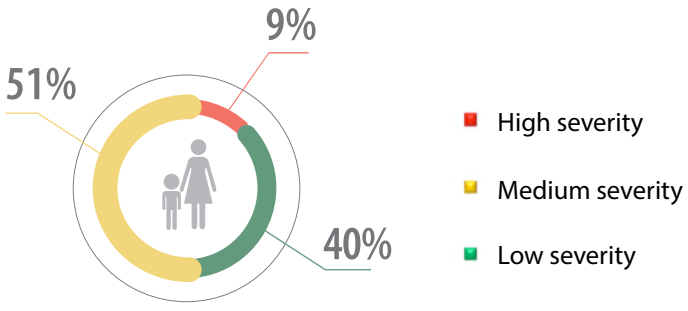

Figure 11. Vulnerability distribution by severity (for systems aimed at individual consumers)

2.3 E-banking System Vulnerabilities for In-House and Commercially Available Systems

Figures 12 and 13 show the percentage of vulnerabilities found in both in-house and commercially developed systems

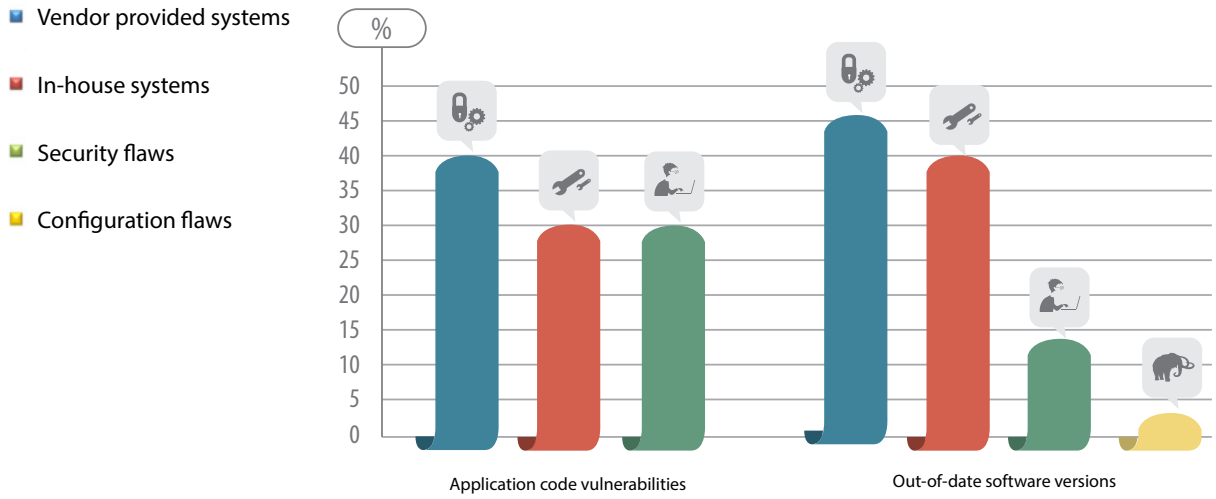

Figure 12. Percentage of various vulnerabilities for in-house and commercially available systems

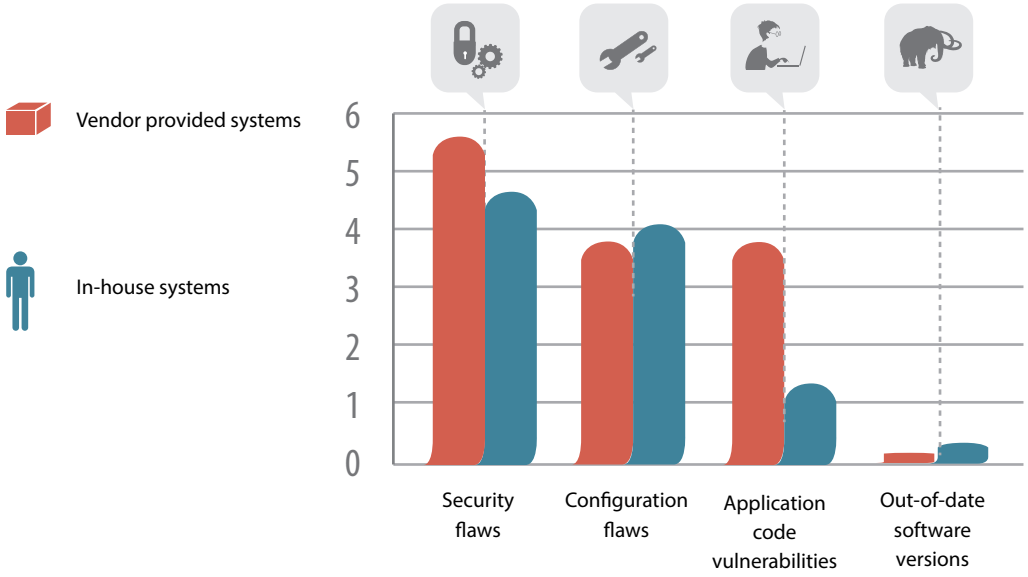

Figure 13. Average number of vulnerabilities for in-house and commercially available systems

It is worth noting that application code vulnerabilities were found to be less common in in-house systems than in commercially available systems. This may be the result of banks that have purchased a system from a third-party vendor relying on that third-party to ensure the integrity and security of their source code. Such applications are usually cross-platform, employ a complex architecture and have many functions making them difficult to be written securely. Figures 14, 15 and 16 provide statistics by severity level of vulnerabilities detected in commercially available and in-house systems.

Figure 14, figure 15 and figure 16 provide statistics by severity level of vulnerabilities detected in vendors' and in-house systems.

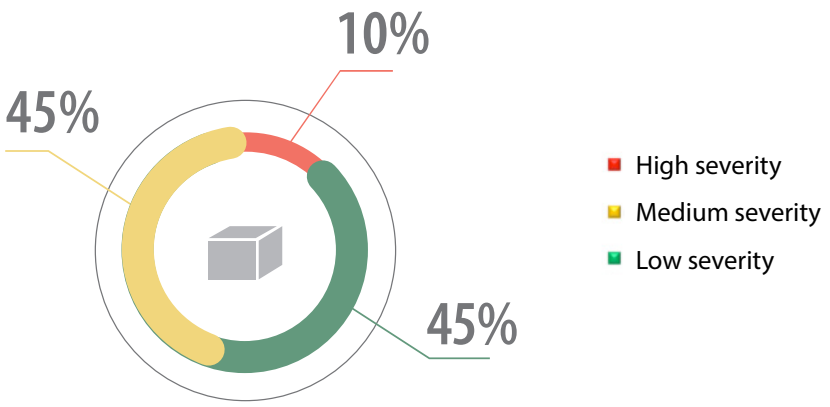

Figure 14. Vulnerability distribution by severity for commercially available systems

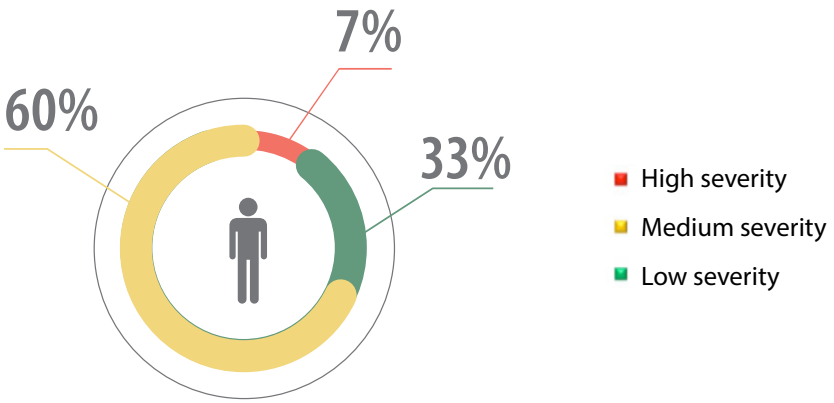

Figure 15. Vulnerability distribution by severity for in-house systems

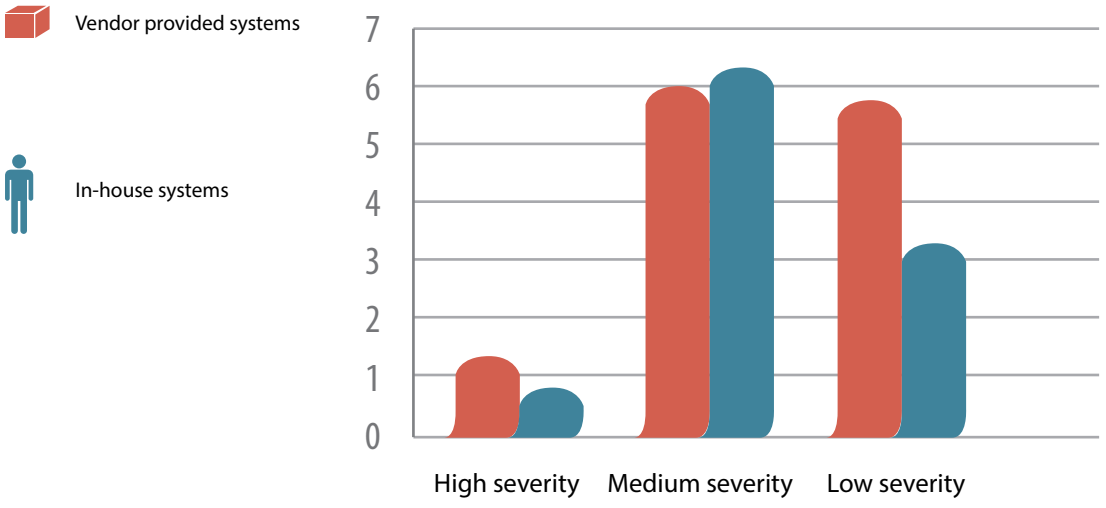

Figure 16. Average number of vulnerabilities in different systems

High-severity vulnerabilities are most common in off-the-shelf systems. Moreover, no high-severity issues were found in the application source code of systems that were developed in-house (see section 7, page 21).

The results show that purchasing a third-party system does not automatically guarantee good security. We highly recommended that banks take ownership of their security and perform their own system checks before putting any system into production and conduct routine audits of the systems over their useful lives.

2.4 Vulnerabilities of Pre-Production and Production Systems

This section provides the results of security analysis of test and production systems. Figure 17 and figure 18 provide vulnerability statistics.

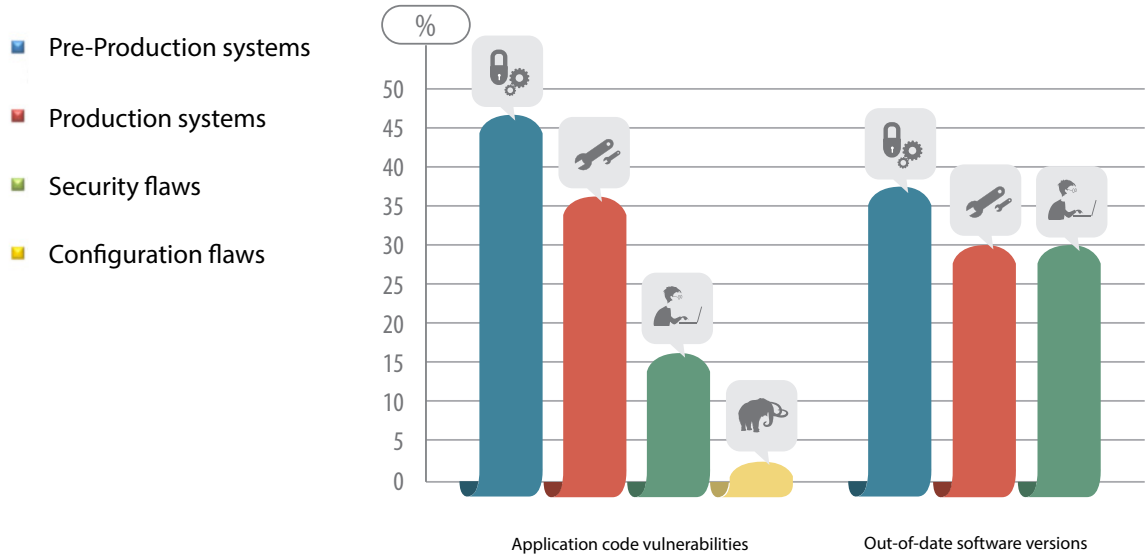

Figure 17. Percentage of various vulnerabilities for test and production systems

According to the research, flaws in security control mechanisms are more common in test/pre-production environments than in production systems (47% vs. 38%); while application source code vulnerabilities are more typical in production systems (31% vs. 16%). See figure 18 below.

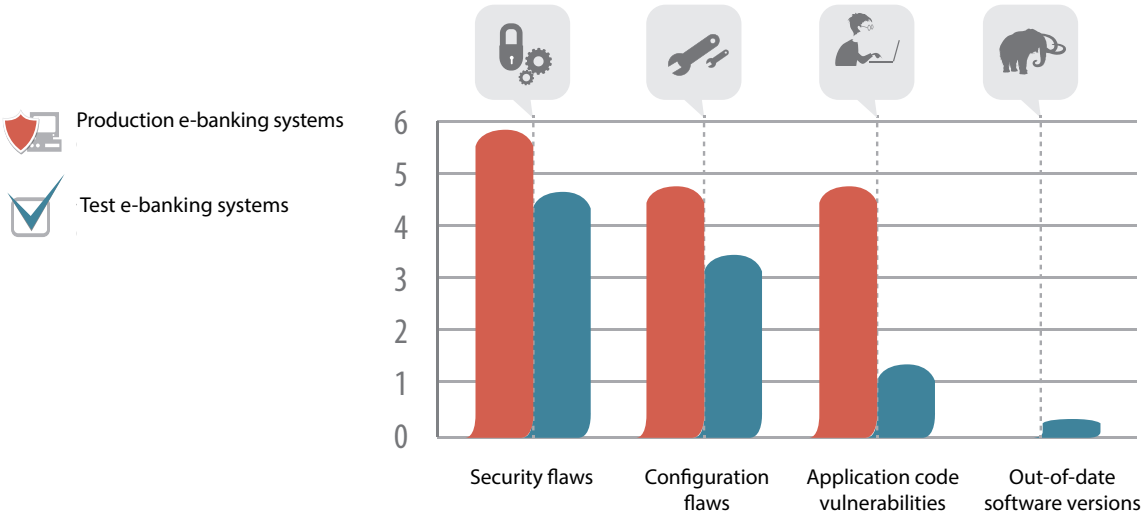

Figure 18. Average number of vulnerabilities in test and production systems

On average, production systems were found to contain more vulnerabilities in all categories, except those related to out-of-date software. This can be attributed to the fact that banks typically don't perform as many security checks after their systems are deployed as they might carry out during pre-production testing. Once systems are in regular use, they are perhaps tested less frequently and if a vulnerability is detected, the complexity of such systems makes it difficult to make the modifications necessary to eliminate the risk.

It should be noted that the only vulnerability related to an out-of-date software version was detected in a pre-production environment, prior to being released to production. This vulnerability is of high severity and can allow an attacker using malware to obtain full control over the system.

Figures 19, 20 and 21 provide statistics on the vulnerabilities detected in test and production systems by severity.

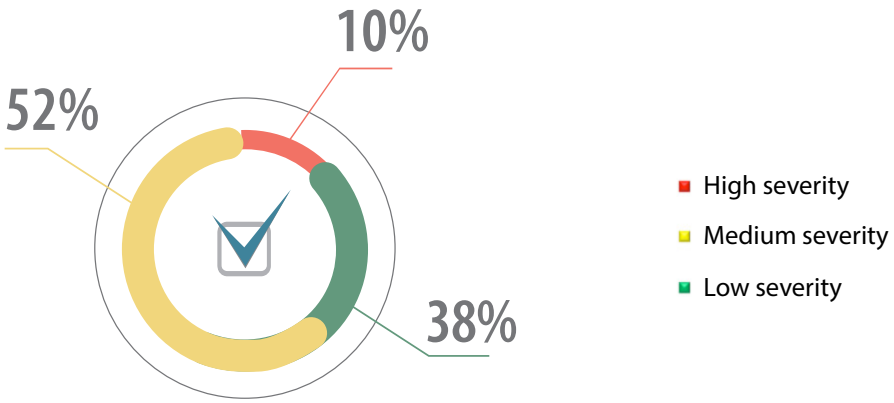

Figure 19. Vulnerability distribution by severity for test systems

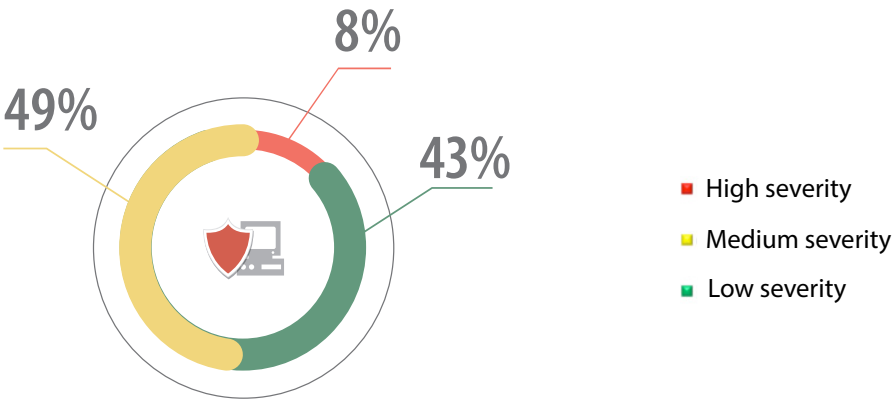

Figure 20. Vulnerability distribution by severity for production systems

The percentage of vulnerabilities detected of different severity levels was almost the same for test and production systems. However, when comparing the average number of vulnerabilities of different severity levels in the two types of systems, it becomes evident that production systems are more vulnerable.

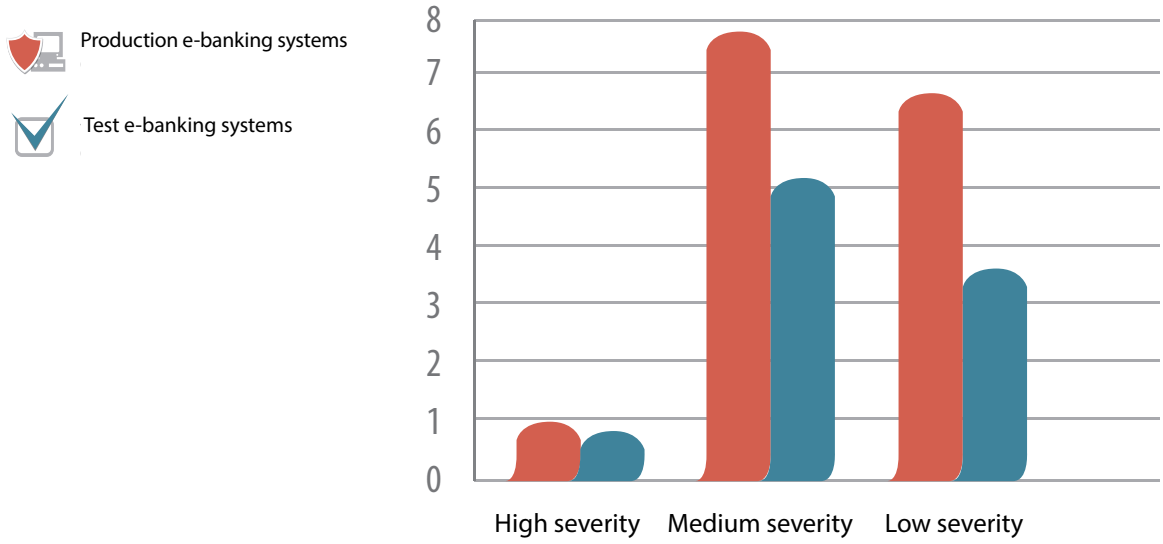

Figure 21. Average number of vulnerabilities of different severity levels in test and production systems

Vulnerabilities of high, medium, and low severity are all more common in systems that have been in general use for some time.

Simply doing a security audit on e-banking systems, when first deployed, is not enough. In order to assure security, regular, ongoing system checks should be performed.

3. THE MOST CRITICAL VULNERABILITIES

The diagram in figure 22 provides a list of all the high-severity vulnerabilities detected in e-banking systems during this research. It provides the ratio of systems with critical vulnerabilities to the total number of the assessed systems.

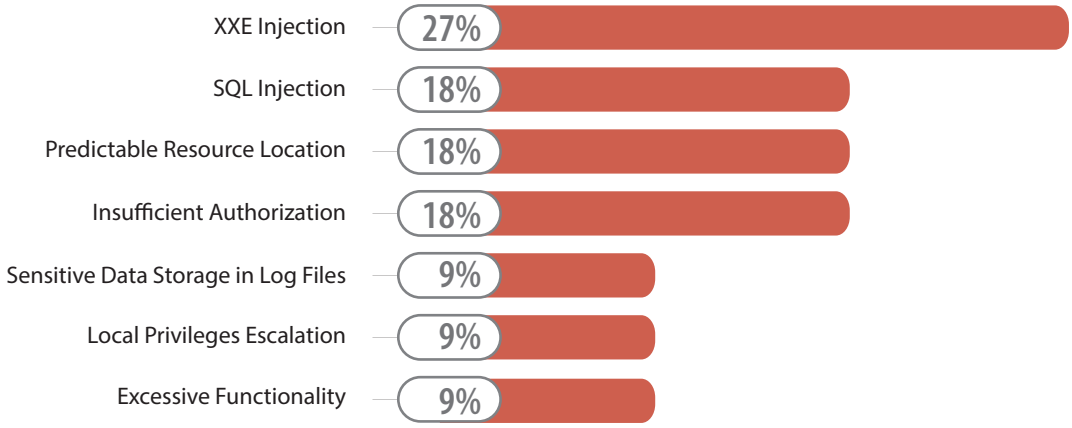

Figure 22. The most critical vulnerabilities found in e-banking systems

In all, high severity vulnerabilities were detected in half of the systems tested. Some examples of how these vulnerabilities could be exploited in e-banking are provided below.

3.1 Excessive Functionality

The web administration console, of one of the e-banking systems, allowed a user to execute arbitrary code, which is an example of excessive functionality. An attacker could access an administration panel by exploiting authentication flaws and then execute arbitrary commands on the server from this screen. By exploiting this vulnerability combined with escalating user privileges, we were able to gain full control over the system.

3.2 XML External Entity (XXE) Injection

XXE Injection is a vulnerability which allows a malware user to obtain the contents of server files. This vulnerability results from improper user data checks by an application – it allows a malware user to conduct an attack aimed at the request logic change by means of XML Injection. Moreover, this vulnerability allows an attacker to execute SMB and HTTP requests in the local network of the attacked server. It may result in the leakage of sensitive data or allow the attacker to obtain web application source code, configuration files, and other important system data.

Exploitation of this vulnerability and the vulnerability Predictable Resource Location in one of the assessed systems allowed us to access e-banking system logs. These log files contained unencrypted information about UIDs and passwords which could then be used to execute fraudulent transactions. In addition, we discovered that information about one-time passwords sent to users via SMS was being backed up unencrypted in this log file and would also be available to a would-be intruder.

3.3 SQL Injection

The vulnerability is exploited by SQL injection and tracking of page content changes. This vulnerability results from improper user data checks by an application – it allows a malware user to conduct an attack aimed at the database request logic change by means of SQL Injection. In the case of a successful attack, a malware user can perform unauthorized actions including:

- Determining DBMS versions
- Obtaining UIDs and passwords
- Obtaining one-time passwords
- Creating and modifying payment data and other information

Exploitation of this vulnerability in one of the systems allowed us to work within the database of an e-banking system with application rights. This means we could perform any transaction without needing user credentials of any kind. It is important to note that no attacks of this kind were carried out during our research, but we were able to prove that such an attack could be performed without setting off the anti-fraud systems, since database content can be changed by the application in a routine way.

4. IDENTIFICATION FLAWS

More than 60% of the e-banking systems we studied had at least one of the following user identification flaws:

- Predictable UID format
- Disclosure of system UID data

These faults allow a malware user to list registered UIDs and conduct attacks aimed at accessing the e-banking system by impersonating a legitimate user (e.g., a brute force attack). Despite the low severity of such vulnerabilities, if it is possible for an attacker to obtain UIDs and use them on a system that also has the detected authentication flaws (see sections 5 and 6), the attacker can obtain unauthorized access to user accounts and perform fraudulent transactions. Moreover, allowing access to UIDs can allow a malware user to trigger a denial of service attack, if the e-banking system locks user accounts after several failed login attempts.

Figure 23 provides statistics on the types of identification flaws found in the systems tested, with a differentiation made between in-house developed systems and those purchased from recognized vendors of e-banking software.

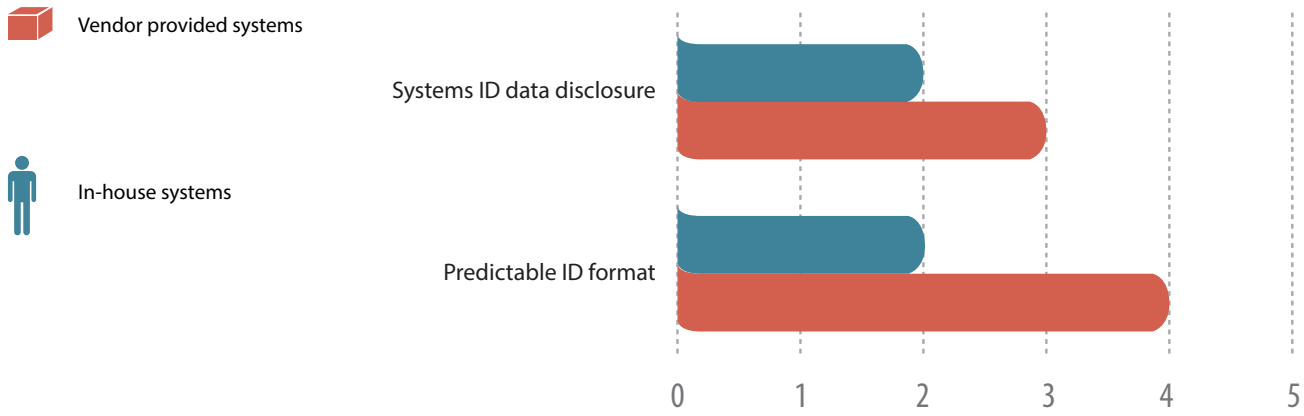

Figure 23. Identification flaws found in vendor provided systems vs. in-house developed systems

4.1 Predictable ID Format

The most common identification flaw found in e-banking systems was a predictable format of account identifiers. This vulnerability was detected in more than half of the evaluated systems (see figure 6).

If an attacker knows a few system IDs (obtained by checking a transaction receipt or having brute forced the system), the ID format may allow the attacker to predict other IDs within the system if they can figure out the algorithm for generating them. The most common identifiers are as follows:

- IDs consisting of digits only (e.g., 344985127) were detected in 27% of cases.
- IDs consisting of digits and a permanent postfix or prefix (e.g., 211113AA, 1231254) were also detected in 27% of cases.

A unique client identifier (UCI), which is not very long (7 characters), is frequently used as a digital identifier. Such identifiers can be easily brute forced.

It is worth noting that 37% of the systems tested used ID formats that were difficult to brute force. ID analysis was not carried out for one of the test systems because the identification mechanism had not been determined by the owner at the time of testing.

Figure 24 provides overall statistics for at risk user identifiers used in the e-banking systems tested.

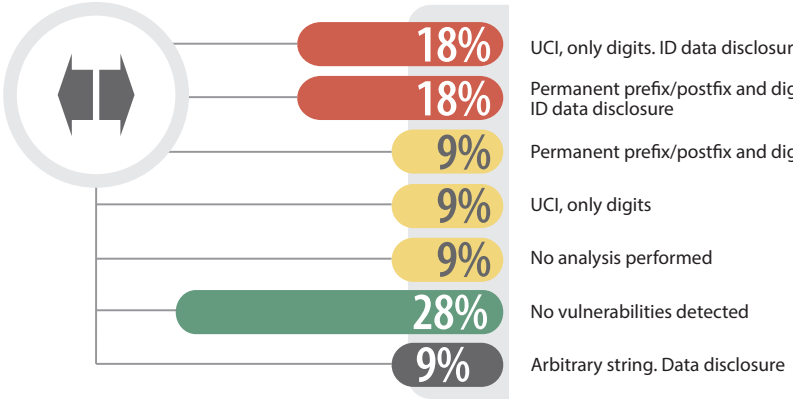

Figure 24. Identification flaws (% of systems where each weakness was found)

To avoid possible attacks resulting from the aforementioned flaws, it is recommended that banks add randomly-generated characters to all user identifiers, for instance, 1234567-Tn4 or 1234568-h2S. This added complexity will make brute force attacks more difficult. In cases where users are allowed to create their own identifiers, it is recommended that systems prevent users from using common dictionary combinations, as their ID (such as QWERTY).

4.2 ID Information Disclosure

Another flaw typically associated with user verification is the fact that UIDs can be disclosed as part of the authentication process. This type of flaw was detected in 45% of the systems we tested. This vulnerability results when it is possible for an attacker to determine user account information based on server responses.

Information disclosure is common for new user registration scripts or password change scripts, when an e-banking system delivers different results for authorized and unauthorized users.

Listing 1 below provides an example of a server response with data disclosure.

```
HTTP/1.1 200 OK

***

<Message>1|The name already exists in the system. Specify another name please.</Message >
```

Listing 1. Information disclosure in a server response

Figure 24 provides ID data disclosure statistics for the e-banking systems we tested. It should be noted that 36% of these systems exhibited both ID information disclosure and a predictable ID format.

5. AUTHENTICATION FLAWS

For the majority of the systems (82%), users were authenticated by an ID and password upon logon. Multi-factor authentication, which requires an additional hardware token (or any other mechanism in addition to a user ID and password), was used only in two systems. These were both systems which aimed at business customers which required the end-user to download an element of the application to their device.

The majority of the authentication flaws found in e-banking systems can be divided into the following categories:

- Weak password policies
- Inadequate protections from brute force attacks
- Lack of multi-factor (strong) authentication

Vulnerabilities connected with authentication flaws such as inadequate protections from brute force attacks and weak password policies were the most commonly-found system vulnerabilities.

Figure 25 shows the number of authentication flaws.

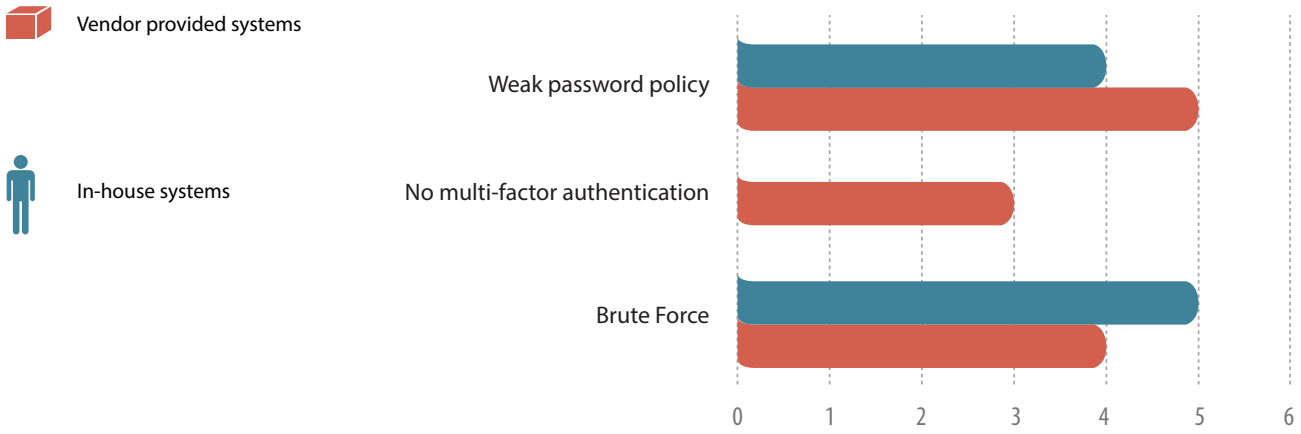

Figure 25. Authentication flaws found in different e-banking systems

5.1 Password Policy Flaws

The most common password policy weaknesses are listed below:

- Incorrectly configured password length (passwords of less than 8 characters can be used)
- Allowing dictionary passwords (e.g., 111111, P@ssw0rd) to be used
- Ability to use previous passwords

These flaws, together with deficient protection from brute force attacks, would allow attackers to exploit user accounts and gain access to the banks' systems. Given that the authentication process is the first barrier entry, password policies should be

considered very carefully. The possibility of choosing a short or dictionary password should be eliminated. Password age should also be limited and a user should not be able to choose a new password which repeats a previous one.

Figure 26 provides the ratio of systems with different password policy flaws to the total number of systems.

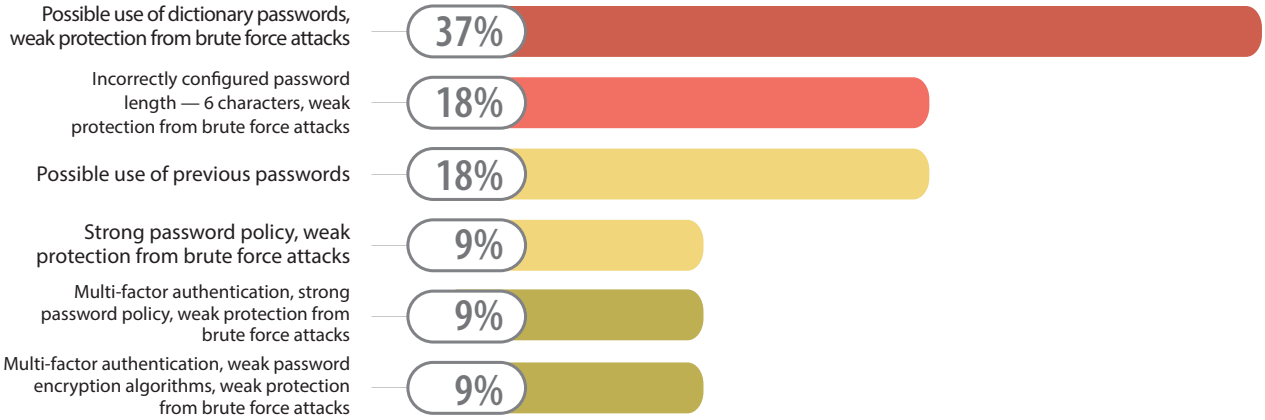

Figure 26. Percentage of systems with different authentication flaws

5.2 Inadequate Protection from Brute Force Attacks

When it comes to protecting themselves from brute force attacks, banks usually use temporal or permanent account blocking after several failed login attempts. However, this mechanism does not completely protect them from brute force attacks. If it is possible to set a dictionary password, an attacker can brute force these despite the limit on login attempts. Moreover, incorrect locking may result in denial of service to legitimate system users.

Figure 26 (above) provides the percentage of password policy flaws and brute force security mechanism flaws in the systems we tested.

Leading banks are increasingly using a system known as Completely Automatic Public Turing test to tell Computers and Humans Apart (CAPTCHA). This method asks users to input a series of characters that are displayed on screen. If incorrect credentials are entered several times, the system locks the account, thus obstructing a malware user of IDs as well as password brute force attacks. The systems we have identified in figure 26 as lacking adequate protection from brute force attacks were either not using CAPTCHA at all, or were using a version of CAPTCHA that contained vulnerabilities.

Use of CAPTCHA together with a strict password policy will significantly strengthen a system's resistance to brute force attacks and we recommend its use in all financial applications. If blocking is used, banks should take into account factors such as the time interval between consecutive login attempts, resource IP address and not only passwords but IDs as a target of brute force attacks as well.

5.3 No Multi-Factor Authentication

To access their accounts, most of the systems tested asked users to provide only their ID and password. Use of a strict password policy and effective protection from brute force attacks together with multi-factor authentication before a transaction

is completed usually ensures a good level of security. However, our tests showed that several systems used multi-factor authentication neither at the user login stage nor for re-verification that the logged-in user was legitimate prior to authorizing transaction. In this case lack of multi-factor authentication was considered as a vulnerability because such poor authentication mechanisms leave an e-banking system at significant risk.

In general, it is recommended that even systems without the first two flaws described in this section should incorporate multi-factor authentication. This would reduce the risk of unauthorized access to user accounts, which usually contain personal and sensitive information. Moreover, if an attacker has user access, they can not only try to bypass authorization and carry out transactions on behalf of a user, but detect and exploit server component vulnerabilities as well (e.g., XXE Injection and SQL Injection).

6. AUTHORIZATION AND TRANSACTION PROTECTION FLAWS

Authorization in all of the e-banking systems we tested was performed by means of user sessions. Some systems use additional parameters such as a UID or unique request token in addition to the session ID for authorization purposes. A session ID had sufficient entropy in all of the systems to obstruct ID brute force attacks. However, the sessions of several systems were not strongly secured from hijacking and further exploitation:

- A session identifier was not linked to an IP address or to a client's browser in 27% of cases. In 18% of cases, it was linked only to an IP address.
- One in three systems allowed more than one user to simultaneously login to the same account.
- Data needed for authorization was transferred insecurely in two systems – in POST and GET parameters (such data could be cached by browsers or hijacked on external resources).

In some cases, the properties secure and HttpOnly were not set for cookie parameters containing session IDs. This makes attacks against user sessions possible (see section 8). In addition, data in many systems was transferred in the clear text and could be hijacked by a malware user.

Multi-factor authentication with one-time passwords (OTP) was used in the majority of the systems to protect at least some types of transactions. However, in some systems there was no multi-factor authentication at all, or it was incorrectly configured. And some 18% of the systems used multi-factor authentication, but only required it for a limited set of transaction types.

According to the research, the most commonly-found authorization and transaction protection flaws in e-banking systems are as follows:

- Insufficient authorization
- Complete failure to implement multi-factor authentication for transactions
- Failure to implement multi-factor authentication on the full range of transaction types

Figure 27 and figure 28 show the percentage of authorization flaws.

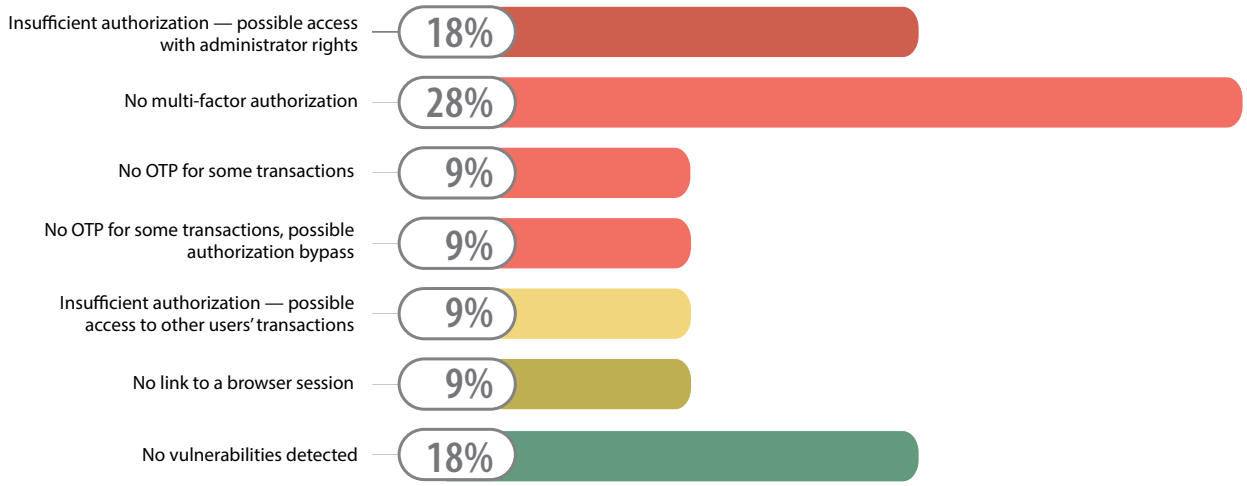

Figure 27. Percentage of systems tested which exhibited each type of authorization and transaction protection flaw

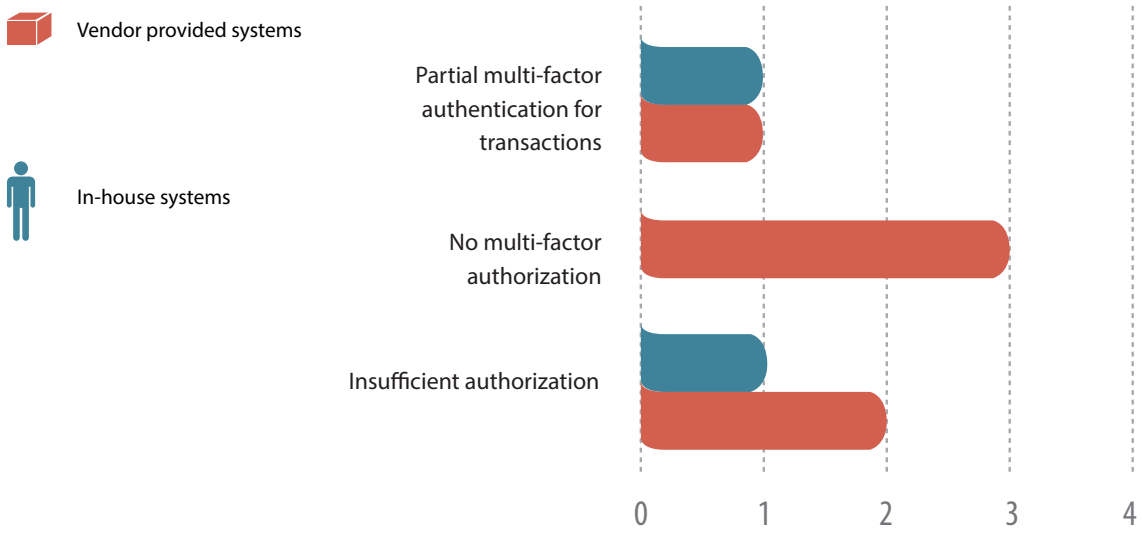

Figure 28. Authorization and transaction protection flaws found in different e-banking systems

6.1 Insufficient Authorization

Insufficient authorization is the most severe authorization vulnerability we found during our research and was present in one out of three systems. By exploiting this weakness, an attacker could access a system with administrator privileges. In fact, this attack was successful against 18% of the systems tested (see figure 28) and in some cases we found that administrative access had been left completely open. This vulnerability results from incorrectly configuring authorization within administration scripts. It allows unauthorized users to access administration system scripts and control key elements (e.g. viewing or deleting data).

To ensure adequate security, administrative access must always be restricted to authorized users to ensure attackers are not able to use application control

functions and elements (including system commands). For example, where this vulnerability exists, an unauthorized user could call system functions such as debugging information output redirection; which would allow a malware user to read a complete history of all system transactions. When exploited together with other vulnerabilities (e.g., excessive functionality), this weakness can result in severe consequences, including malware users gaining full control over the e-banking system. Application source code analysis is required to eliminate this vulnerability from systems.

6.2 No Multi-Factor Authentication for Transactions

Using multi-factor authentication, like one-time passwords (OTP), requires users to provide an additional credential before executing a transaction – increasing the protection against unauthorized access and unauthorized transactions. However, our research shows a third of the e-banking systems we tested did not use any form of multi-factor authentication. In these cases, an attacker could easily carry out fraudulent transactions by first stealing user credentials through brute force or session hijacking.

6.3 Multi-Factor Authentication for Only a Limited Set of Transactions

If multi-factor authentication is used, but not required for all transactions, then it might as well not be used at all. If multi-factor authentication is not required for some transaction types, an attacker can exploit those particular types of transactions to withdraw money from user bank accounts.

7. WEB APPLICATION CODE VULNERABILITIES

Of the e-banking systems we tested, 82% were found to have vulnerabilities in web application source code. Figure 29 illustrates the percentage of vulnerable systems distributed by the severity level: 37% of the systems contained high-severity vulnerabilities, while 45% had medium and low-severity vulnerabilities.

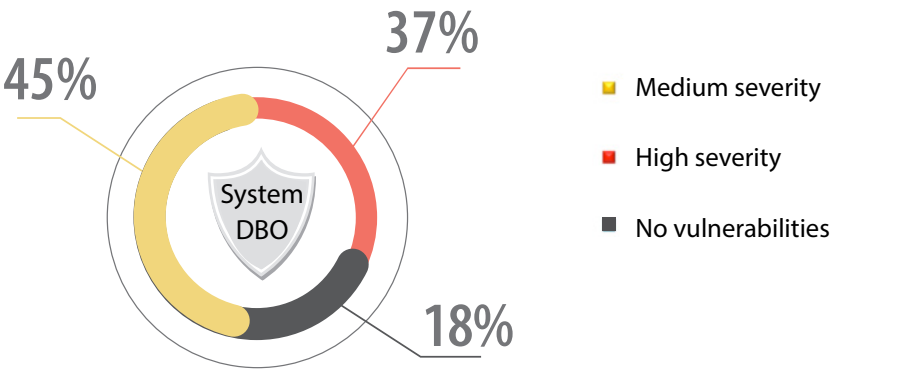

Figure 29. The percentage of e-banking systems with vulnerabilities of various severities

In total, 16% of all the application vulnerabilities we detected were classified as being of high-severity (see figure 30 below).

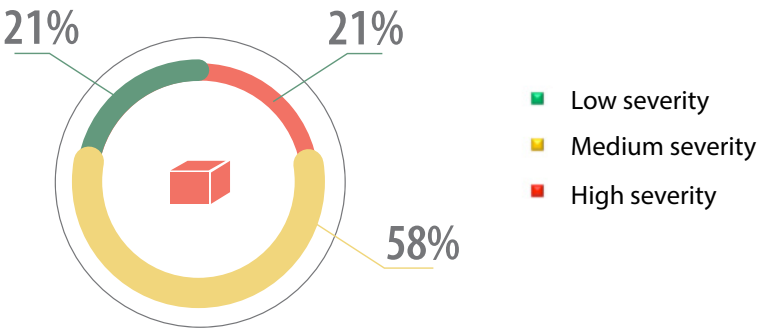

Figure 30. Total detected application vulnerabilities categorized by severity

Figure 31 provides the total number of the detected application vulnerabilities.

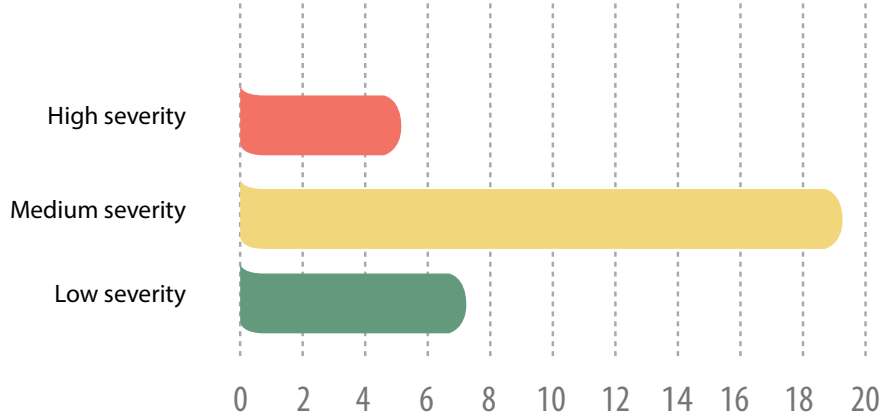

Figure 31. Number of detected application source code vulnerabilities

7.1 Application Vulnerabilities of In-House and Commercially Available Systems

Both applications developed in-house and those provided "off-the-shelf" by third-party vendors were found to have vulnerabilities. The research showed that in-house e-banking systems had less vulnerabilities overall and had no vulnerabilities of high-severity (see figure 32).

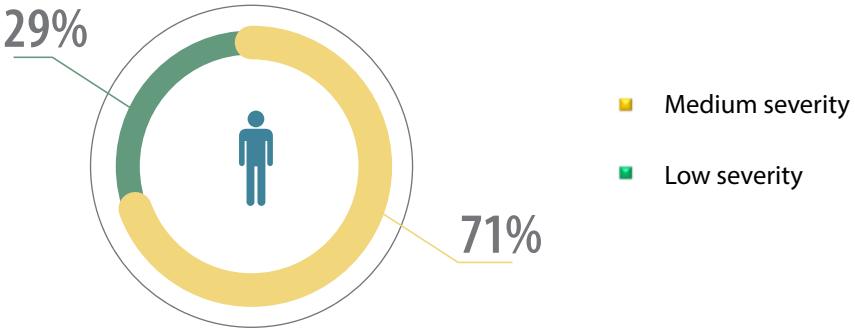

Figure 32. Severity of vulnerabilities found in systems developed in-house

All the high-severity vulnerabilities found were detected in commercially available e-banking systems. Figure 33 gives the ratio of such system vulnerabilities.

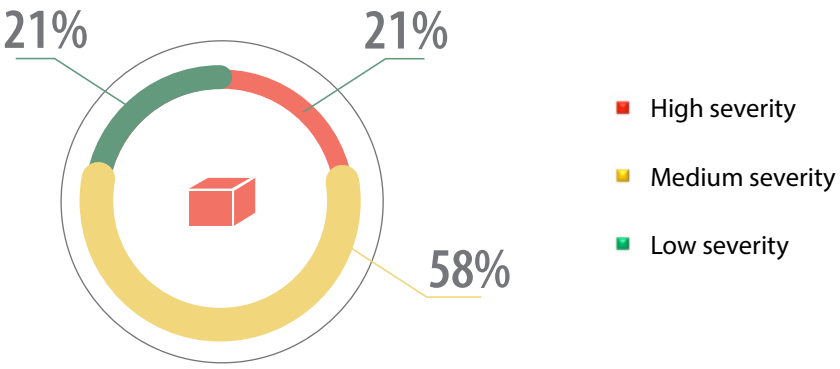

Figure 33. Vulnerability ratio of commercially available systems

Section 3 (page 14) provides a description of the most severe vulnerabilities found in this area.

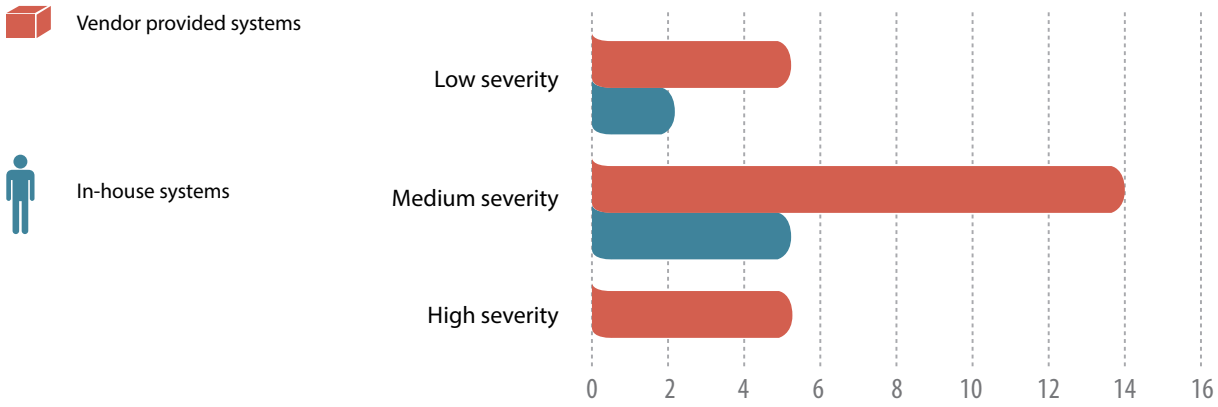

Figure 34. Total number of application vulnerabilities detected

7.2 The Most Common Web Application Vulnerabilities

Figure 35 demonstrates the number of e-banking systems, which were found to contain application source code vulnerabilities. Figure 36 provides the ratio of vulnerabilities detected in off-the-shelf applications to those found for in-house systems.

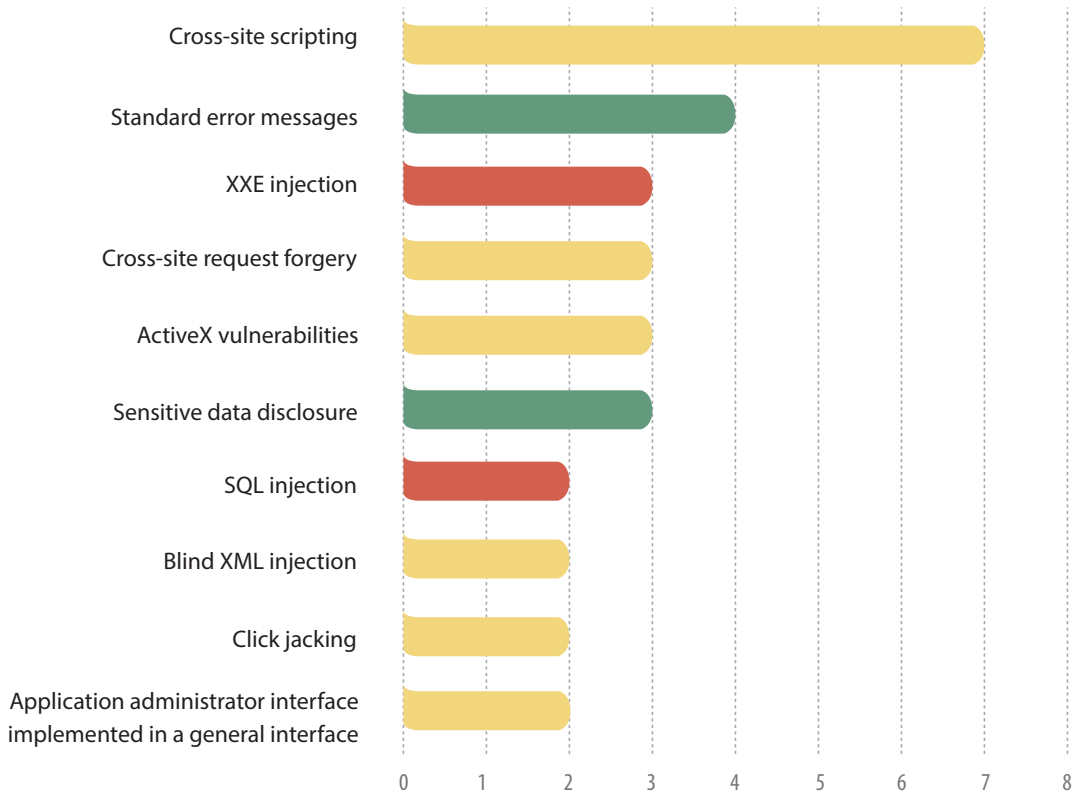

Figure 35. The number of systems found to have application vulnerabilities

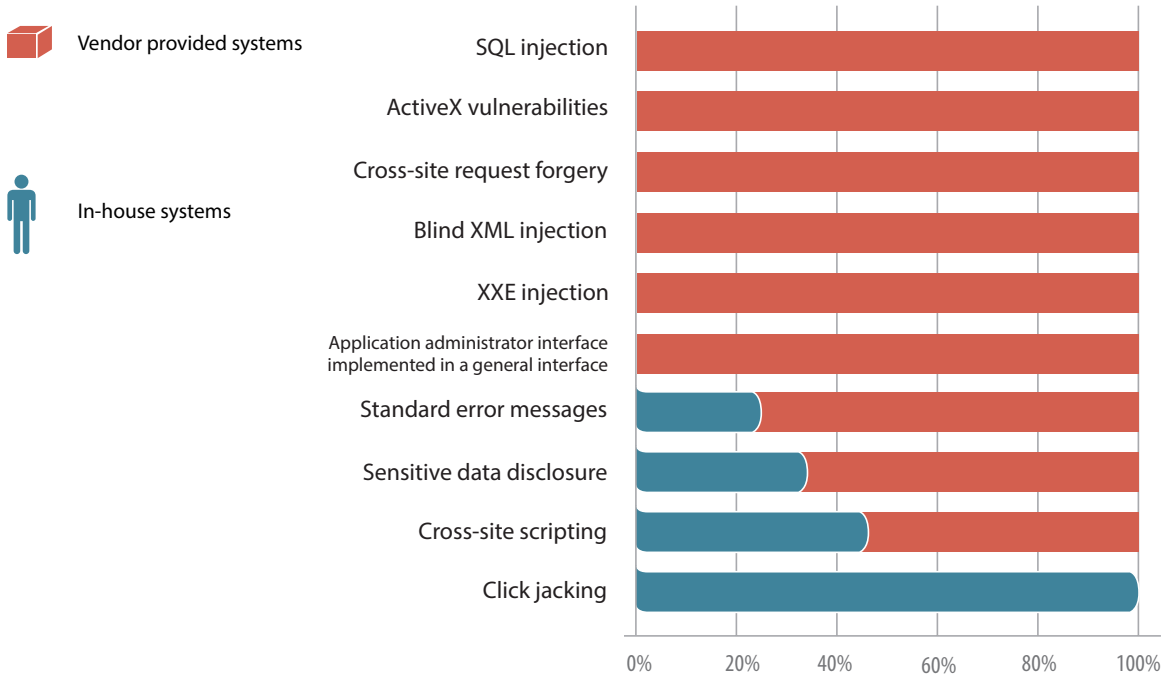

Figure 36. Application code vulnerabilities found in commercial systems versus those developed in-house

Multiple Cross-Site Scripting vulnerabilities were detected in more than a half of the e-banking systems analyzed (see figure 6). Therefore, we can say that half of the systems tested are open to client-side attacks. This vulnerability results from improper user data checks being performed by a web application. It allows attackers to inject arbitrary HTML tags into a user's browser including JavaScript scripts. Using this method, attackers can conduct client-side attacks using phishing to hijack user credentials and gain unauthorized system access. Once in they can propagate malware via the computers of e-banking system users. In one of the systems which failed to use multi-factor authentication, this vulnerability could have been exploited to perform mass money withdrawals from user accounts.

ActiveX is another client-side vulnerability. Such vulnerabilities were detected in all the three of the systems that required end-users to download elements of the application to their own device. These were provided by third-party vendors. These vulnerabilities can be exploited through attacks including Cross-Site Scripting and XXE Injection. Exploiting different ActiveX components, an attacker can either read or write an arbitrary file on a user's computer.

To reduce the risks of web application vulnerabilities, it is recommended that developers implement secure development procedures including Systems Development Lifecycle (SDLC) practices, analyze applications regularly (ideally in source code), and eliminate all vulnerabilities – including application code weaknesses immediately. If a financial institution is using an off-the-shelf system, we recommended the use of a Web Application Firewall, while you are waiting for a fix from the third-party vendor.

8. CONFIGURATION FLAWS

This category of flaws is caused by the incorrect configuration of operating systems, DBMS, web servers and web application components. One third of all the vulnerabilities detected in this study were of this type (see figure 5, page 6); with all of the systems tested having at least one configuration vulnerability. In 18% of these cases, configuration flaws were classified as posing a high security risk (see figure 37).

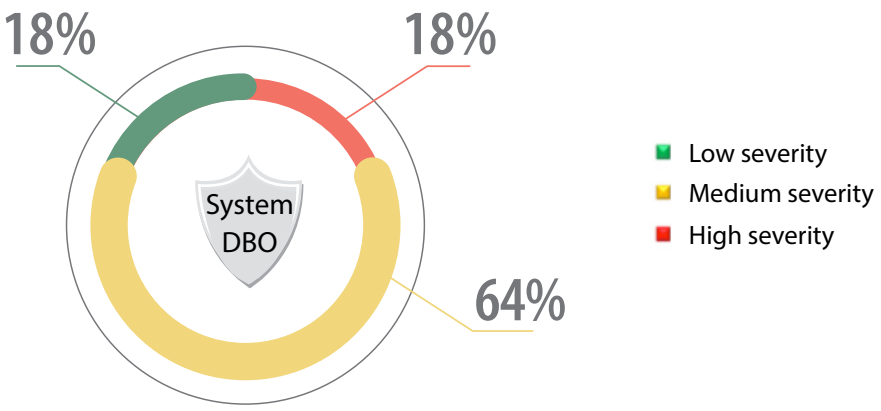

Figure 37. The percentage of systems with configuration flaws (by maximum severity)

Although 18% of the systems tested were at high-risk from configuration vulnerabilities, in terms of actual numbers, only two high-severity vulnerabilities were found. One of them was found in a commercially available system, while the other was discovered one developed in-house.

These two high-risk flaws represent 4% of the total configuration vulnerabilities found. The percentage of medium and low-severity vulnerabilities of this type is almost identical: 49% and 47% respectively (see figure 38). Figure 39 provides a breakdown of the numbers of detected vulnerabilities of different severity levels.

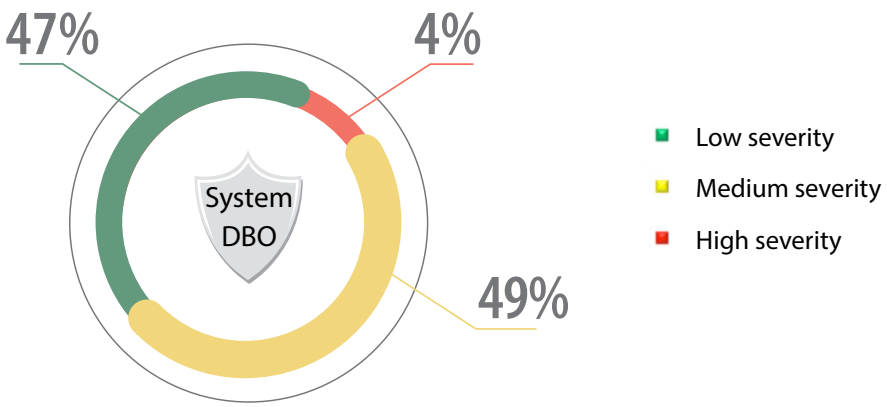

Figure 38. Configuration vulnerabilities found by severity level

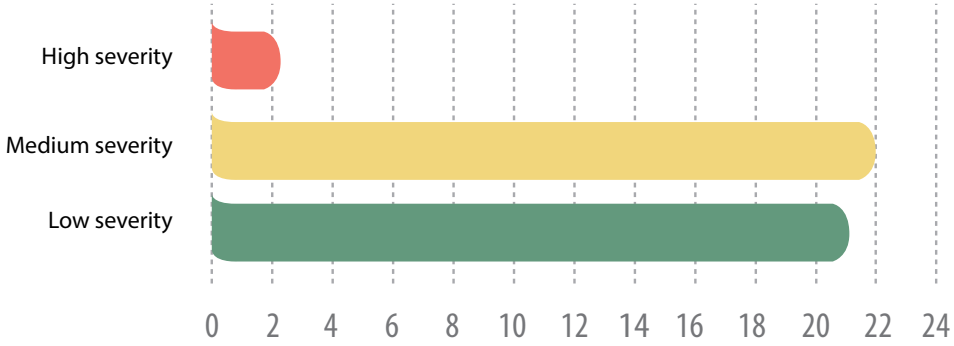

Figure 39. Number of configuration vulnerabilities found

The most severe vulnerabilities of this type are excessive functionality, predictable resource location and insecure protocol use. Section 3 of this report (page 14) describes these in more detail.

In addition, the following configuration flaws were also found to be very common:

- Unencrypted data transfer (found in 45% of systems tested)
- Insecure configuration of cookie parameters (found in 36% of systems tested)

One example of this type of flaw is the failure to properly configure the secure property for the cookie parameter, which transfers a session ID. If this vulnerability is present, a browser can transfer the parameter not only via HTTPS but also via the HTTP protocol as well. If insecure data transfer protocols or weak encryption algorithms are used, an attacker can hijack user data. It should be noted that the

majority of the systems we found using insecure data transfer protocols were test systems where secure data transfer was to be implemented prior to production.

8.1 Configuration Flaws of In-House and Commercially Available Systems

The majority of the configuration vulnerabilities (58%) found in commercial systems were of the low-severity type. Whereas, the most common class of configuration vulnerabilities found for in-house systems were medium-severity (62%). The percentage of high-severity configuration flaws is almost the same no matter the systems' origin (4% and 5% respectively). Figures 40 and 41 demonstrate the configuration vulnerabilities found based on the origin of the systems.

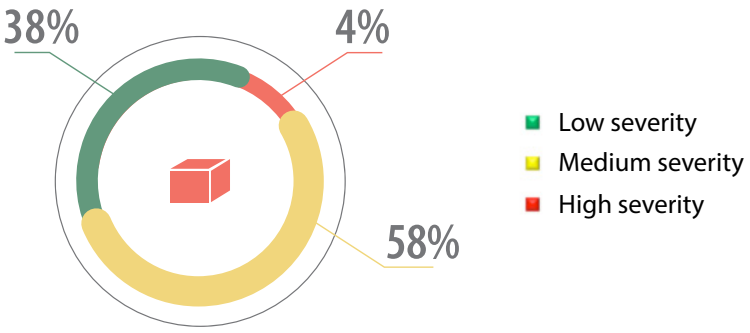

Figure 40. Configuration vulnerabilities in the commercially available systems

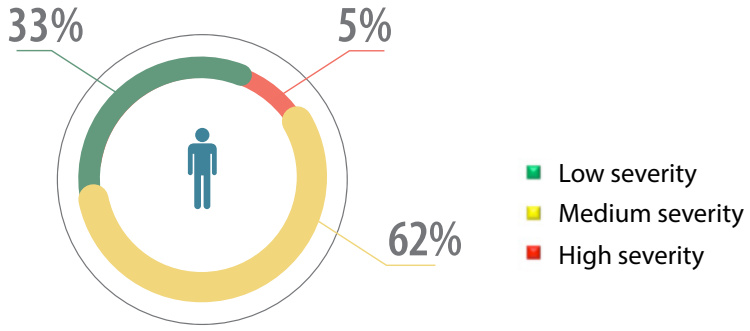

Figure 41. Configuration vulnerabilities in the in-house systems

Figure 42 provides the quantitative ratio of configuration vulnerabilities of each severity level detected in the systems we tested. The results suggest the security levels of the in-house developed and commercial systems are almost identical from the point of view of e-banking systems configuration.

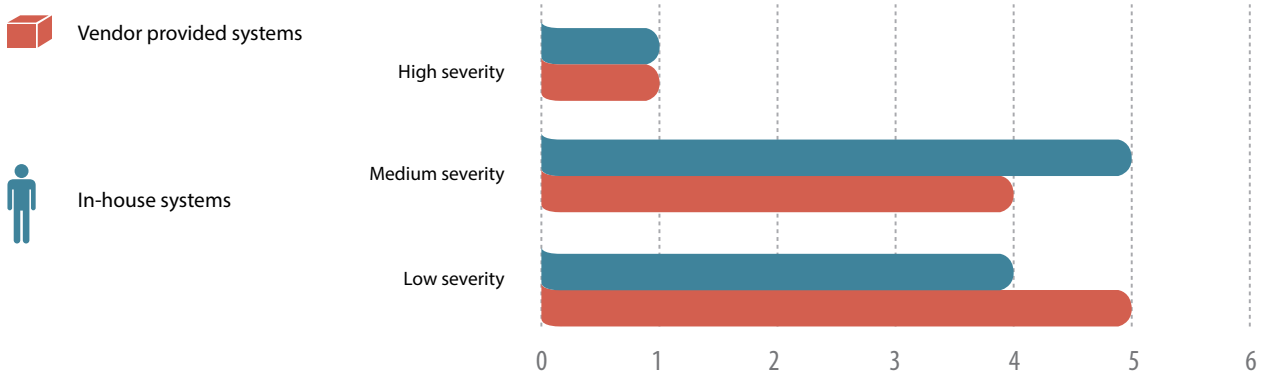

Figure 42. Configuration flaws detected in in-house vs. commercially available systems.

9. OUT-OF-DATE SOFTWARE

While carrying out security analysis of their e-banking systems, banks usually limit themselves to searching vulnerabilities that are caused by configuration settings, errors in the application code and logical flaws in the way the application operates. In our experience, system owners typically believe that regular updates of the e-banking application will be carried out and will fix all known vulnerabilities associated with the use of old software and DBMS versions.

However, systems with certain application vulnerabilities will lead to disclosure of information about the versions of software components used within the system, and by exploiting this disclosed information, we have been able to prove that many of the e-banking systems have numerous additional vulnerabilities in their OS and DBMS caused by out of date software. This demonstrates that financial institutions may not be as diligent at updating the software in their systems as they profess to be. It is important to perform security assessments of all server components of an e-banking system in order to detect the presence of these vulnerabilities. During such an investigation, specialists can accurately reveal versions of installed software, define whether current security updates are installed or not, and assess the feasibility of exploiting vulnerabilities to gain control over the system.

E-banking systems, like any other information system, are prone to vulnerabilities associated with the use of obsolete versions of software. In most cases, an intruder can only use these defects after exploiting application vulnerabilities, but vulnerable versions of OS and DBMS can lead to significant extension of an intruder's privileges – even allowing them to gain full control over an e-banking system and access to a bank's internal network. We recommend banks perform security assessments on server components regularly and update software promptly, in order to prevent an intruder from exploiting known vulnerabilities.

10. OTHER FLAWS

The e-banking systems analyzed in the course of this research had a series of other significant flaws. For instance, log files containing the following sensitive information were detected in one of the systems:

- Authentication data (UID and password hash)
- SMS with a code needed to confirm a transaction
- Installation path (can be used by an attacker to exploit some vulnerabilities)
- Unencrypted password (the password is stored in the log file when it is changed according to a certain script)

This data can be used by an attacker to obtain unauthorized access to a system and subsequently to transfer money or carry out other unauthorized actions.

Primary account numbers (PAN) were poorly masked in some systems we tested. In some cases a PAN was hardly disguised at all or was masked in a web interface but displayed in the HTML code of the page. This flaw violates the requirements of the

Payment Card Industry Data Security Standard (PCI DSS) and can be exploited using malware to target the bank's customer accounts. Three out of eleven systems tested used no PAN masking at all.

A Denial of Service (DoS) attack is one of the most severe threats related to the Internet. In the case of e-banking systems, this type of attack can be aimed at rendering the entire system unavailable or at locking-out individual users, with obviously financial and reputational implications for the bank concerned. By making use of a predictable UID format, an attacker is able to lock user accounts by simulating multiple failed login attempts.

CONCLUSION

This research has demonstrated the scale of vulnerabilities present in current e-banking systems. The results we obtained prove once again that financial institutions must perform their own security analysis, even if the system was purchased off-the-shelf. Moreover, institutions must pay attention to the proper configurations of systems security mechanisms, especially to those related to authentication and authorization. They must also ensure a high-level of quality control related to the development of web applications. If a system is already in use, banks are strongly advised to check the configuration settings of each of its components, and to update the software regularly.

When using commercially available systems, we recommend using a Web Application Firewall until the third-party vendor releases an update which prevents attackers from exploiting application source code vulnerabilities.

An e-banking system, as with any other information system, requires a comprehensive approach to security at all the stages of its lifecycle. Secure development implementation and regular control of e-banking system security will reduce the risk of unauthorized system access and protect the bank – and its clients – from fraudulent activities.

EBANK_WP_A4.ENG.0015.01.DEC.29.2014