nccgroup
freedom from doubt

An NCC Group Publication

The Why Behind Web Application Penetration Test Prerequisites

– how you can help us help you secure your web apps

Prepared by:

Jerome Smith

Contents

Document History

Issue No.	Issue Date	Change Description
1.0	18/06/2015	Approved for public release

1 Introduction

Before a web application penetration test is scheduled to start, the company performing the test will contact the client with a set of prerequisites; that is, a list of considerations and configurations that are required before the test can begin. Sadly, the importance of these can be lost in amongst the frenzy of ensuring the system is ready on time, or perhaps because the items listed don't seem necessary. This whitepaper aims to address the latter, providing an explanation of the why behind the prerequisites, together with some general pre-test points for consideration. The focus of this paper is on web application rather than infrastructure tests, as these tend to cause more exceptions.

The paper is aimed at anyone who is charged with preparing for a web application penetration test, from project managers to developers, and as such it is written for both technical and non-technical readers. With this in mind, a test against a fictitious travel booking service is used to illustrate some of the points. Obviously this paper cannot be prescriptive since applications, their use cases and their environments differ wildly, but it presents the main issues that can become obstacles to testing. Understanding and addressing these beforehand will return the best value for money from a penetration test.

2 Introducing our example

The scenarios in this whitepaper are based on a fictitious business travel operator *BusiTravel* at *www.busitravel.local*[1]. Organisations with staff that travel frequently can register with *BusiTravel* to receive discounts on business travel. The typical use case is that an in-house travel coordinator has an account and books flights, car rentals and hotels as needed. The coordinator can set up an account for the traveller, who can then view details of their various bookings. The traveller can also store and update data, such as their passport number and driver's licence details, which the coordinator may need for making reservations. Within *BusiTravel* itself, administrators can create and maintain any type of user. We therefore have three types of user:

User Type	Function
Traveller	Can view their own bookings; Can access and update their personal details
Coordinator	Can make bookings on behalf of Travellers; Can read but not edit Travellers' details
Administrator	Can read and edit bookings; Can create and edit users of any type

3 Accounts

Having established the URL of the site to be tested (*www.busitravel.local* in our example), this section tackles the question of accounts.

3.1 Why do I have to configure more than one account?

High up on the list of prerequisites will be a request for two (or more) accounts per user role, the reasons for which are discussed below.

[1] No similarity to any real organisation is intended and the .local subdomain is used to reinforce this (http://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml)

3.1.1 Horizontal privilege escalation

The primary reason for consultants requiring multiple accounts is based on a class of vulnerability called “horizontal privilege escalation”. This may appear to be something of a contradiction in terms (after all, something horizontal doesn’t imply escalation) so some readers might prefer the phrase “horizontal authorisation flaws”. Either way, it’s an easy concept to understand when we consider the *BusiTravel* website: it’s clear that two Travellers shouldn’t be able to see each other’s details. If Traveller A can exploit a flaw to see the personal details of Traveller B, for example, that’s an authorisation failure.

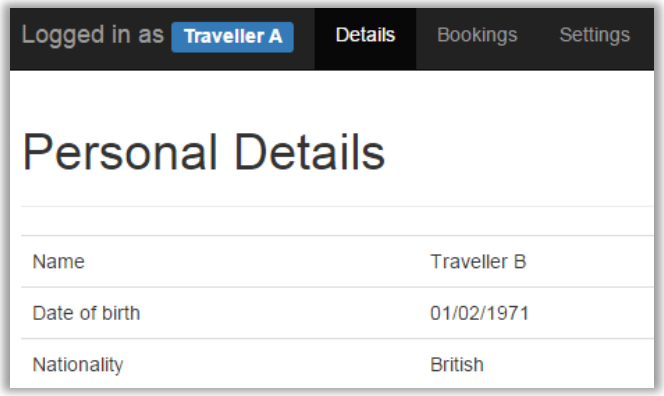


Figure 1: Traveller A can see Traveller B's details

Because Traveller A can access something that should have been forbidden, Traveller A is said to have escalated privileges; but because the victim is in the same user group as the attacker, it’s classified as “horizontal”. In order to test this authorisation boundary the consultant will require two Traveller accounts – and the same principle applies to all user levels.

3.1.2 Multiple consultants

Sometimes a penetration test is scheduled with more than one consultant. If the consultants are using the same accounts, it’s possible that they could interfere with each other’s testing. This very much depends on the application and may be avoidable with coordination between consultants, although this does incur something of an overhead.

In some cases, however, it may be essential to create extra accounts in proportion to the number of consultants. For example, if Traveller A is logged in and someone else attempts to log in as Traveller A, the application may terminate the first session in order to prevent concurrent logins[2].

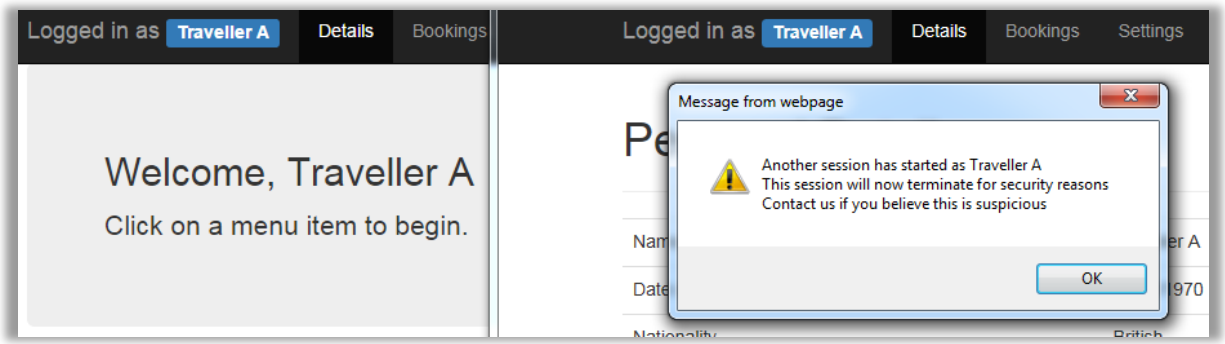


Figure 2: A new login as Traveller A (left) has terminated the current session (right)

[2] For more information refer to https://www.owasp.org/index.php/Session_Management_Cheat_Sheet.

Another scenario is that the application is “stateful”, in that it is programmed to track the progress of a user and anything unusual results in the user being returned to the home page or even logged off. In both of these scenarios it is clear that two or more consultants cannot make use of the same account.

3.1.3 Locked or corrupted accounts

Any consultant with experience will know that functions that could risk locking out an account should not be tested until the end of the assessment. This is because the process of unlocking an account can cause a delay, as well as an extra burden on administrators. Despite this, it’s possible that an account can become locked by accident – for example, when an application is programmed to respond to malicious input by locking the offending account, but the consultant is initially unaware of this. Alternatively, an account may be rendered unusable because it becomes corrupted as a result of testing activity or a bug in the application.

If any such scenario occurs, a consultant may have to wait for an account to be reinstated; but if another account is available then there need not be a delay. Therefore, in the process of configuring accounts before the test, adding one more may be worthwhile to safeguard against this contingency. It’s also advisable to notify the consultants up front of any account lockout policy in place, so that it can be factored into the assessment.

3.2 Isn’t it cheating to give you an admin account?

Not necessarily, and this subsection goes on to consider the general case of applications that support multiple user levels. Users at the same level (or, if you prefer, role) have the same set of permissions assigned to them. In the case of our example website, recall that *BusiTravel* has three user levels: Traveller, Coordinator and Administrator.

3.2.1 There’s no such thing as cheating

Before continuing, it’s important to realise that, in the context of a penetration test, there really is no such thing as cheating. This is because the consultant’s goal is to find security issues before they are exploited in a genuine attack. Unlike a motivated attacker, a consultant is limited for time and thus any requests that seem unrealistic within a real attack scenario are made simply to expedite the process of testing and analysis. An experienced consultant will always report the conditions required for an attack to succeed so that the risks can be properly understood and evaluated by the client who commissioned the test.

3.2.2 Vertical privilege escalation

In contrast to horizontal privilege escalation (discussed in section 3.1.1), the term “vertical privilege escalation” should be easier to understand; in fact instead of a contradiction in terms we now seem to have a tautology! The vertical type of privilege escalation is what would perhaps be considered more typical – when a user gains privileges that were not assigned to them.

For example, imagine that a Traveller was using the same computer that a Coordinator had previously used. While typing in the URL for *BusiTravel*, the browser’s address bar automatically makes some suggestions, which includes a page that the Traveller did not know existed:

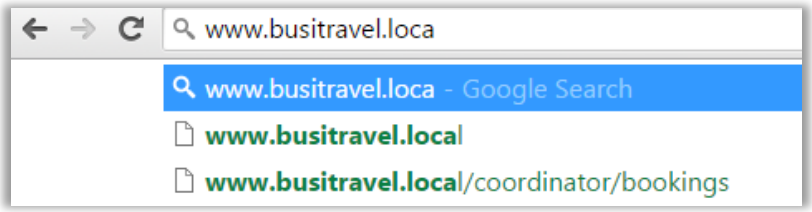


Figure 3: The browser suggests a page unknown to the user

Having logged in, the Traveller finds that the page *www.busitravel.local/coordinator/bookings*, which suggests that it is only for the eyes of users in the Coordinator role, is in fact accessible:

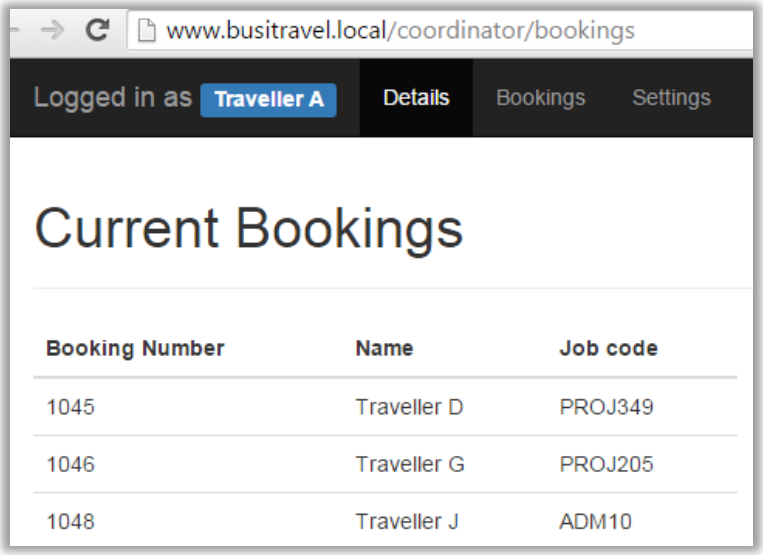


Figure 4: A Traveller user can see bookings for the entire company

Checking for cases of vertical privilege escalation is the reason that consultants ask for credentials at different user levels. Without access to the Coordinator account, the above flaw could have been missed during a penetration test.

Note that in some cases an authorisation flaw that allows escalation somewhere between horizontal and vertical could exist (although we don't propose to coin the term "diagonal"!). One user level may be just as important as another, but with both carrying a different set of permissions (which may even overlap). If a user from one level obtains unauthorised permissions, it may be reported as "escalation" simply because they have gained extra rights.

3.2.3 But our normal users don't have admin access

That's exactly what the test will check. Indeed, users outside of *BusiTravel* itself do not have administrative access – or at least they shouldn't. But what stops a curious Traveller from navigating to the page `/admin/`? And what if the page that was returned looked like this…?

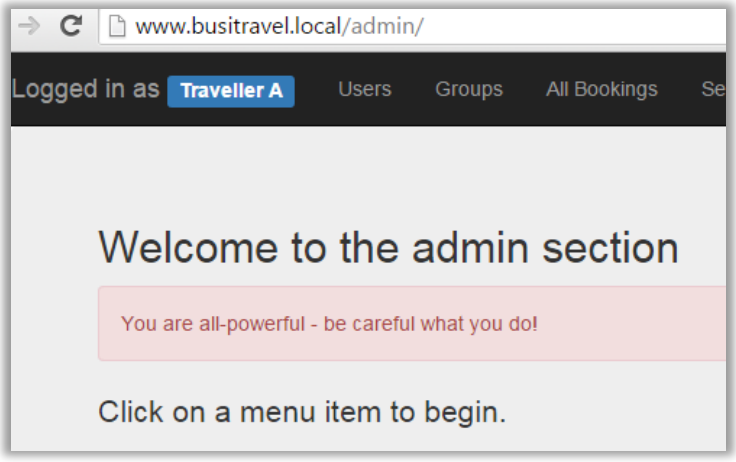


Figure 5: A Traveller user can access an administrative area

Even if the URL is not so easy to guess, other clues that point to the locations of administrative pages may exist. In the final analysis, security should not be left to obscurity: authorisation checks

should be performed wherever necessary. To make best use of limited time, providing admin access speeds up the process of testing such authorisation boundaries.

3.2.4 Sources and sinks

Another reason for requesting accounts at different levels is that some attacks may be *delivered* in the context of one user level but *executed* in the context of another. In other words, the "source" and "sink" of the attack straddle an authorisation boundary. Let's return to the *BusiTravel* site for an example. When a user changes their password, a log event is created that only administrators can see:

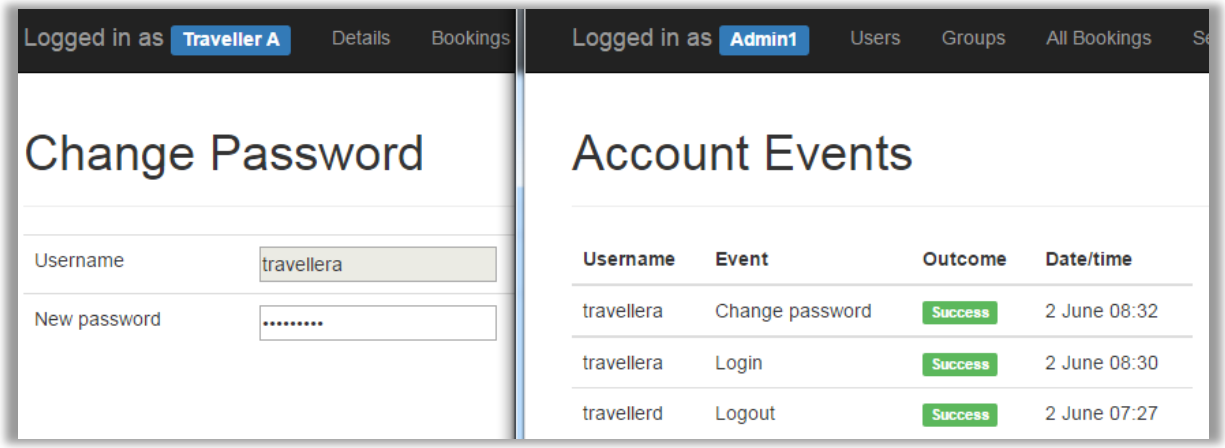


Figure 6: A change password event creates a log entry

Although the username field is disabled on the Change Password page, a savvy low-level user can re-enable it for editing and, as a result, have control over a portion of a log page seen by administrators:

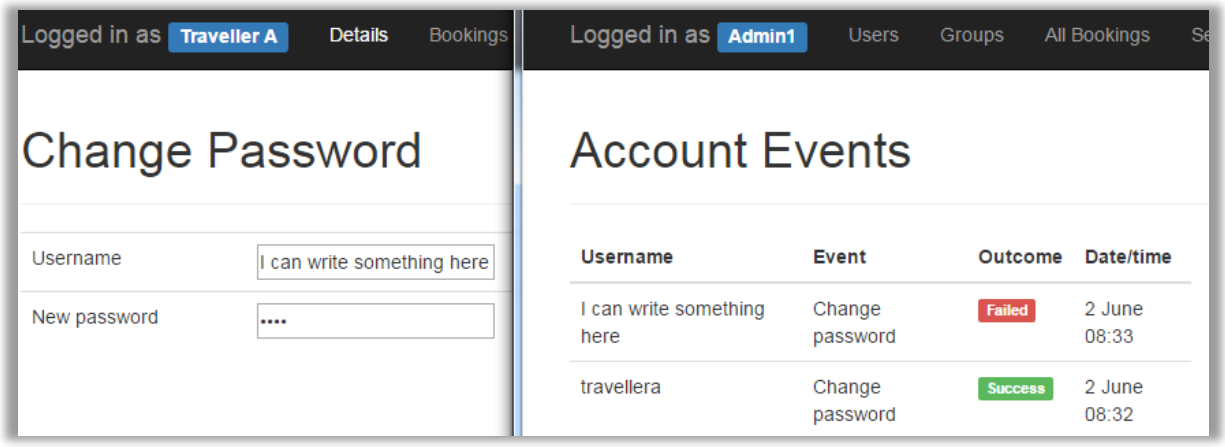


Figure 7: Traveller A can write to a page that administrators view

While this is a mere defacement, it may be possible to engineer the username in such a way that actual code runs in the context of the administrator's session[3]. This is a dangerous flaw and, in this instance, could only be reported if admin access was available during testing.

[3] This would be a case of "stored" (or "persistent") cross-site scripting (XSS); for more information refer to https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_%28XSS%29.

3.2.5 Our admin accounts are just too precious

Sometimes an organisation may decide not to provide administrative access, especially in the case of live systems that perform sensitive actions. That's fine, but it's important to realise this limitation, and it is recommended that the same codebase be tested in a non-live environment to give a level of assurance about the security of that area.

3.2.6 We don't have user levels or roles

Some applications are built with highly granular access controls, where the concept of user "levels" or "roles" does not exist. Instead, each user is configured with any number of permissions. This may look something like the following:

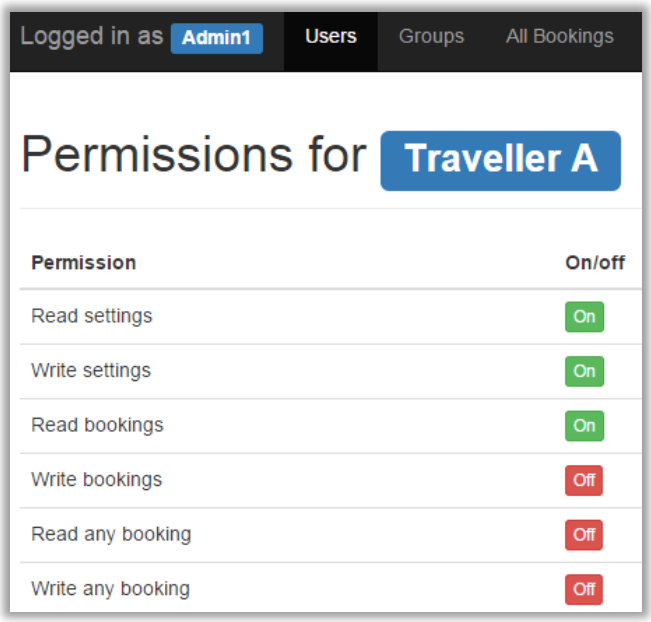


Figure 8: Granular permissions for users

In this case, it is recommended that a limited set of users be configured, representing the most typical profiles, and with at least one user including the most sensitive permissions.

A similar recommendation follows for applications that support a large number of different user levels or roles. Picking, say, three levels to represent a low-, a medium- and a high-privileged user will allow good initial coverage of the authorisation boundaries.

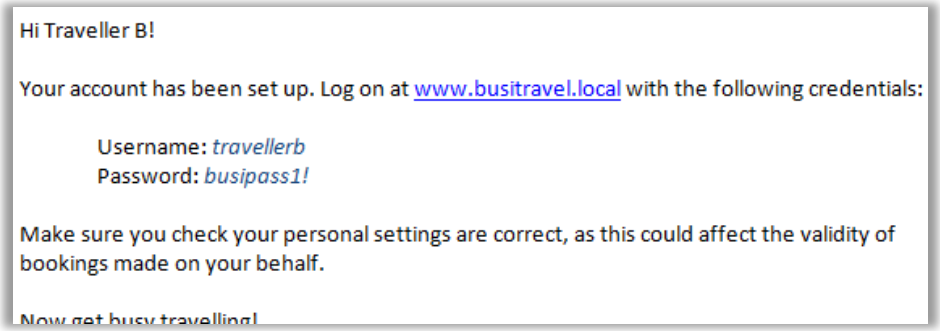
3.3 Why do I have to configure more than one company?

Some sites are used by multiple customers, so it's important that segregation at the cross-organisation boundary is working correctly. Returning to the *BusiTravel* example, imagine that a Coordinator from one company could see the personal details of Travellers from a different company. Or what if a Traveller could access pages from a Coordinator belonging to another company? Thus both horizontal and vertical boundaries should be tested in relation to crossing this organisational boundary. When applicable, another account per user level should be configured under a second organisation.

3.4 Account creation

Where possible, it is recommended that the passwords for all of the test accounts are created using exactly the same process used to set up new accounts for each of the user levels in scope. If the usual steps are bypassed because "it's just a test", the risk remains that a flaw in the normal

procedure will not be discovered. For example, when a Traveller account within *BusiTravel* is set up, the user receives an email:


Hi Traveller B!

Your account has been set up. Log on at www.busitravel.local with the following credentials:

Username: *travellerb*
Password: *busipass1!*

Make sure you check your personal settings are correct, as this could affect the validity of bookings made on your behalf.

Now get busy travelling!

Figure 9: Automatic email sent on account creation

When *two* new Traveller accounts are created (for the reasons discussed in section 3.1) the consultant notices that the same default password is set. This could be used by an attacker to compromise other accounts whose usernames are known and whose passwords have yet to be changed[4]. If the *BusiTravel* administrator had created the accounts in a non-standard way (using direct access to the database, for example) this issue could be missed.

Consider, too, if anything more than a username and password is required – that is, a "second factor" authentication token or smart card. This will need to be configured, and possibly sent, in advance of the assessment's start date. Even if the token is software-based, it is recommended that the requirement be raised in advance to allow time for testing this dependency.

4 It's not just accounts

This section describes some examples of other details unrelated to accounts that may have to be pre-configured. Often they can be set up during a test if a hurdle is reached but, in general, the more that can be done in advance the better.

4.1 Email addresses and phone numbers

For some applications, a user's profile will include details such as an email address and a phone number. Sometimes these can't be changed without administrative action but they are a vital part of functions such as account recovery (e.g. a link is emailed in the case of a forgotten password) or authentication (e.g. a login code is sent via SMS). In such cases, the consultant's details should be used instead of a random email address and phone number to which the consultant has no access.

4.2 Discount codes, registration codes (etc.)

Consider any part of the website that requires data that would be unknown to the consultant – for example, discount codes applied to purchases, or codes or other details used for registration. Sorting these out ahead of time will help to prevent delays during the test, ensuring that they are tested first time around.

4.3 Payment cards

If the website allows goods or services to be purchased using a payment card, consider how that should be tested. Usually non-live systems are either not linked up to a payment processor or they make use of the processor's own test payment gateway. In these cases, test payment cards are

[4] There are other issues with this approach too but they would be a distraction at this point.

usually available, either built-in to the application or offered by the provider. In the case of a live system, test cards may still be configured as part of the application, or the company itself may make use of a real payment card with limited funds for testing. Ideally this information would be made available to the consultant, otherwise extra administration may be incurred in arranging refunds, depending on the sums involved and the agreement in place.

5 Pre-populated data

Applications work on data. If no data is available for the user to work on then there's very little to do and consequently very little to test. Therefore penetration tests require applications to contain enough data to allow the features of the website to work in a realistic way.

Sometimes websites allow the user to create data, in which case the consultant can do this – and those parts of the application will be tested along the way. But there are other times when pre-populated data – that is, data configured *in advance* of the test – is at the very least helpful, if not mandatory. Should this be missing and become something that must be addressed during the test, valuable time may be lost. The remainder of this section discusses scenarios in which configuring pre-populated data may be necessary or advisable.

5.1 Privileges

The privileges of the accounts supplied may not allow the consultant to create data.

5.2 Sources

The application may consume data from external sources outside the control of the consultant – and perhaps even outside the control of the client who commissioned the test. In non-production environments, external data sources may not even be available. Another potential problem is that data is created too infrequently.

5.3 Time/complexity

Even if the consultant can create data, there may be cases when it would take a disproportionate amount of time for the consultant to do this. The process of creating a realistic data set may be just too lengthy or complex, perhaps because the application is used within an esoteric business area (on which point, refer also to section 8). While the part of the application that is used to generate data may be important to test for security issues, it may be too onerous to include in every test of the system.

5.4 A case for heterogeneous data

When pre-populated data is required or deemed to be worthwhile, it's also helpful to configure different data between different accounts, as this makes it easier for the consultant to assess authorisation boundaries. Imagine that a problem exists within *BusiTravel* such that Traveller A can see Traveller B's bookings, but the test data was set up in such a way that Traveller B's profile was simply copied from Traveller A. While testing for an authorisation flaw in this area of the application, the consultant sees the following:

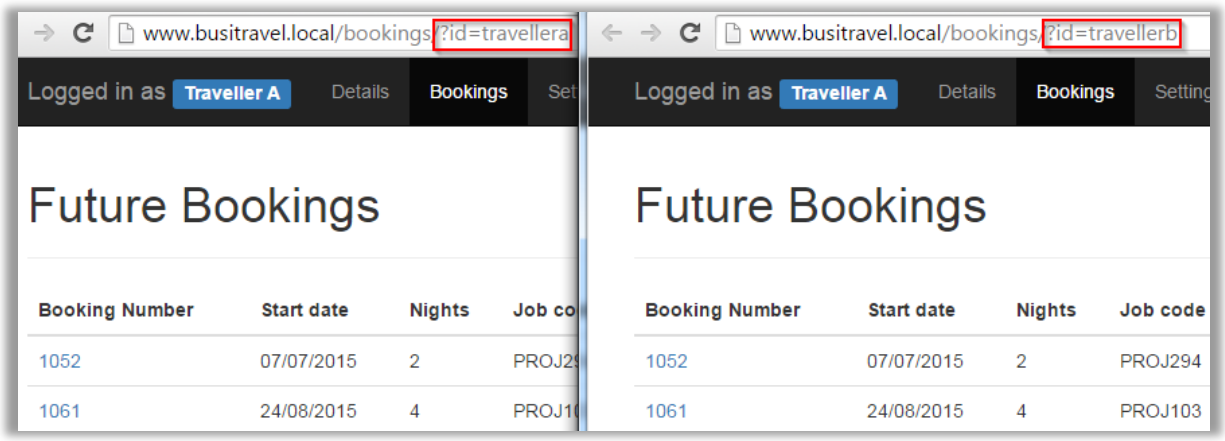


Figure 10: An attempt by Traveller A to see Traveller B's bookings

From this it looks like everything is fine – after all, no error was returned, and it appears that Traveller A's bookings were returned in both cases. However, we know that this conclusion could be a false negative, because Traveller B's bookings are identical to Traveller A's. Providing the consultant has access to both accounts, this conclusion should not be drawn from the available evidence, but if the consultant cannot amend the data for one of the Travellers (or only one Traveller account has been supplied) then a definitive test cannot be run, and the issue could be missed. The difference between the accounts' data doesn't have to be great, just enough to notice.

6 End-user technical requirements

Applications that do not have a wide user base, especially those used internally, sometimes have specific demands of the client machines connecting to them. Some examples follow:

- Software, e.g. Silverlight or a bespoke media player;
- Software version, e.g. Internet Explorer 7 or Java 6;
- Software settings, e.g. the site must be added to Internet Explorer's Trusted Zone;
- User authentication includes "something the user has", such as a client certificate or hardware token.

Missing any of these could lead to lost time while a consultant troubleshoots why the application does not work properly from their laptop or, in the last example, waits for a second authentication factor to be configured.

7 Environment

A penetration consultant should ask for the operational status of the environment hosting the application – and usually the main question is whether or not the environment is live. The consultant will use this information to adjust the nature of the tests being carried out, in order to minimise the risk of any adverse effect on the system and its users.

The use of third parties can change the effective status of an environment. For example, imagine that *BusiTravel* hosts a test environment within Microsoft's Azure cloud service. While *BusiTravel* may consider it to be a mere test site, Microsoft certainly would not!

7.1 Third parties

If third parties are used to host the environment or are part of the application itself, they must be considered during the scoping of the test, and appropriate authorisation must be in place before the test can begin. Since this can take days to arrange, it should be initiated as far in advance as possible.

Note that if a penetration test has been scheduled with a number of days for testing distinct from a number of days for reporting, it is recommended that the authorisation covers the total period of time, as testing may occur in the time that has been nominally set aside for reporting. This is because many consultants report as they test or, in the process of reporting, it becomes clear that something must be checked.

8 Workflows and user guides

8.1 Complex business area

The logic of some applications is inherently easy to understand. For our example travel site, it's obvious that one Traveller shouldn't be able to see another Traveller's details. For other sites that occupy specialised areas of business, how the application works and what should and should not be possible may not be so simple for an outsider, such as a consultant, to understand. Organising a demonstration, writing a set of workflows or making documentation accessible will help the consultant not only to navigate the site efficiently but to find logic errors specific to that application.

Such applications also tend to benefit from data being populated in advance (refer to section 5). Realistic data can help to add context to the application as well as assisting the consultant with understanding the risk when security issues are found that relate to data exposure.

8.2 Scope

Some tests have specific scopes, where only a subset of the site's features is under review. Especially for large applications, a workflow to illustrate how to get to those features will help to speed things along. Screenshots are especially useful. It's important that the workflow includes where the scope of testing starts and ends. For example, imagine that new features have been added to the *BusiTravel* site shortly after a full application test, and another test is commissioned to test those new areas. The workflows should cover all the areas that are affected by changes, for which all user levels should be considered. Specifically, recall from section 3.2.4 that a password change causes a log event to be created in an administrative page. A change is implemented to request the user's old password as part of the Change Password page. A workflow to guide the consultant would be:

- Login as a Traveller, go to Settings > Change password
- Login as an Administrator, go to Logs > Account events

In this case, the website is simple enough for consultants to figure out the process for themselves but, for larger applications, adding some directions will be helpful.

A workflow need not take every conceivable path through the application. Just like a general application test, a consultant will assume that all pages and parameters from the "start of scope" to the "end of scope" are to be tested. If applicable, the workflow should indicate exceptions which the consultant should not test.

8.3 Do the workflows both work and flow?

In an ideal world, when sizeable workflows have been newly created, it is worth testing them from beginning to end, preferably by someone other than the person(s) who wrote them. As well as outright errors, the reviewer should try to flag instructions that assume prior knowledge of the application or its business area. Catching problems at this stage will help to prevent delays during the test.

9 Web application firewalls (WAFs)

Web application firewalls (WAFs) are products that look closely at the web traffic sent to an application. Should anything look out of the ordinary or just plain malicious, the WAF can respond in

a predefined way, such as bouncing the user to an error page or terminating the session. While a WAF can provide defence-in-depth for certain classes of application flaw, during an *application* test it is counterproductive to have it running.

9.1 Why should I turn my WAF off, isn't that cheating?

With the WAF on, the penetration test becomes as much a test of the WAF itself as of the application. If the focus of the test is the application then certain application flaws may remain hidden because the WAF blocks attempts to exploit them. While this sounds good, a genuine attacker with more time and resources may be able to ultimately bypass the WAF to exploit the back-end flaw (refer to section 3.2.1 on why "there's no such thing as cheating"). So while a WAF is a useful defensive measure for an application, it should not be seen as a substitute for secure code.

Additionally, an active WAF will add an overhead to the test that would not have been considered at the scoping stage (unless it had been specifically raised at the time). This overhead could result from, for example, a WAF responding aggressively and terminating the consultant's session frequently.

At the same time, testing the effectiveness of a WAF may be important to the business. This can be factored in with a phased test:

- Phase 1: application test with WAF **off** (clarified in section 9.2 below), allowing unfettered access to the site to find application-level issues.
- Phase 2: application test with WAF **on**, repeating the attacks found in phase 1 to see what effect the WAF would have had and whether or not the protection can be bypassed.

Note that a WAF isn't the only product that could interfere with testing – for example, an intrusion detection or prevention system (IDS/IPS) could also spot irregular network traffic that triggers a defensive response. Therefore, security systems at both the network and application level should be considered.

9.2 What is a whitelist?

It is undesirable for any security product to be switched off entirely during an assessment, especially of a live site. Mature security solutions allow exceptions to be configured on the basis of a "whitelist" (usually a set of approved IP addresses), and the consultant will likely suggest this as a solution.

10 On-site testing

Everything covered so far applies equally to web application tests that are conducted at the premises of the client. This is usually arranged because the application is not available externally but sometimes it is desirable that the test team is in close proximity to the developers or other stakeholders. Whatever the reason, some other prerequisites to consider for on-site engagements follow:

- Desk space
- Chair
- Network connectivity (the need for anything other than a standard RJ45 network cable should be raised)
- Notification at reception
- Security pass
- Will the consultant need to present any ID on arrival?
- Are there any processes that must be completed in order for the consultant to use their own laptop or mobile devices? For example, the physical "MAC" address of the network card may be needed in advance in order to add the consultant's laptop to a whitelist of devices approved to connect to the corporate network. It is highly desirable that the consultant's own laptop can be used, as it will have specialist software with which to perform the assessment, as well as a personal repository of tools, scripts and notes.
- If required, a parking space nearby is always appreciated.

11 Contacts

A penetration test should not be thought of as “fire and forget” – different people within the organisation may be required to assist during the test, and their input can make a real difference to its success. Some examples follow:

- Who can the consultant turn to in case of functional problems?
- Who can the consultant turn to with regard to understanding the application logic?
- For an on-site test, who should the consultant ask for at reception? It’s usually a good idea to have two – and to brief them beforehand!
- Who should be informed of any serious security issues found during the test?
- In the case of a lack of response, is an escalation route required?

12 Quick Reference Guide

By way of a conclusion, the most useful way to summarise this whitepaper is to provide a Quick Reference Guide. This single A4 page (overleaf) can be printed and used as a cheat-sheet to help to prepare for web application penetration tests. It does not cover all of the points raised in this whitepaper but should provide a good starting position.

Web Application Penetration Test Prerequisites – a Quick Reference Guide

High-level Area	Considerations	Requirements
Accounts	Which user levels are in scope?	Two accounts per user level. Spare capacity is recommended.
	Does the application support concurrent logins?	Extra accounts may be required depending on the number of consultants.
	Does the application logic tolerate multiple sessions from the same account?	
	Is there an account lockout policy?	Provide the policy up front.
	Does the application support a wide client base made up of several organisations?	Create a second test organisation, within which one additional account per user level should be configured.
	How are accounts at the various levels created normally?	Follow any processes in place to create the test accounts.
	Does a user log in with anything more than a username and password?	Configure and notify in advance.
Non-account details	Can email addresses and phone numbers be changed by the user?	If not, set up accounts with the consultant's details.
	Is there other information the consultant will need in order to make full use of the site but which would be unreasonable or impossible to know?	Provide in advance.
	Does the site involve taking payment?	If possible, consider how this can be tested without the need for real financial transactions.
Application data	What data is required for the application to work normally? Would any of this be impossible or onerous for the consultant to create?	Configure in advance, preferably with differences between accounts where appropriate.
End-user requirements	Does the application require specific software on the user's machine?	Raise any requirements in advance.
Environment	Are third parties involved?	Ensure authorisation is in place.
Workflows	Does the application demand specialist knowledge to use? Are only specific features in scope?	Create and test workflows to assist the consultant with site navigation and use.
Active defensive software	Is a web application firewall (WAF), intrusion detection or prevention system (IDS/IPS) or similar software in use?	Add an exception for the consultant's IP address ranges for the duration of the test.
On-site testing	Is the test on-site?	Consider the logistics of the consultant coming on-site and connecting to the office network.
Contacts	Who could be required to assist?	Brief relevant staff, e.g. those involved in system administration of the site, its business area, or security. Consider how issues should be raised by the consultant and, if necessary, escalated.

