



CENTRE *for* DOCTORAL TRAINING *in*
**CYBER
SECURITY**



CDT Technical Paper

04/14

**Introducing Survey++: An Extensible
Platform for Testing the Usability and
Security of Credential Recovery
Mechanisms**

Vincent Taylor

Introducing Survey++: An Extensible Platform for Testing the Usability and Security of Credential Recovery Mechanisms

Vincent Taylor and Ivan Flechais

Department of Computer Science,
University of Oxford, United Kingdom.
{vincent.taylor, ivan.flechais}@cs.ox.ac.uk

Abstract. Mainstream authentication procedures have usually relied on knowledge factors for determining whether to allow a user access to resources. Typically, a user is challenged to provide a token that they know to prove that they are authorised. In the case that this token (password, passphrase, personal identification number, etc.) is forgotten, the usual approach has been to provide password hints or password reset questions during credential recovery to help ensure a user is who they claim to be before allowing them to reset the token. Survey++ is a platform designed to test the usability and security of credential recovery mechanisms. Survey++ was built to test a credential recovery mechanism that reminds a user of their password by showing them characters from the password, but it can be easily extended to test other recovery mechanisms or authentication procedures.

Keywords: usability testing, password security, password recovery, testing platform

1 Introduction

Survey++ is a platform for usability and security testing of credential recovery mechanisms and authentication processes. It is built around an online survey platform that serves the purpose of distracting users from the main aim of the experiment. To use Survey++, a researcher only needs the email addresses of consenting individuals and a choice of configuration options for the system. Once these options have been set and the survey launched, Survey++ automatically prompts users via email whenever they are required to complete sections of a survey.

Survey++ is able to measure the authentication and credential recovery process because a user is required to set or recall a password (or other token) each time they attempt to access a part of a survey. Extensive (and configurable) behind-the-scenes logging is in place to capture a user's behaviour and input each time they interact with the system. From this logging, a researcher is able to see authentication failures, input timings, how many times password hints were requested, etc.

Survey++ was built to test the usability and security characteristics of a novel credential recovery scheme in which users are reminded of their password by being shown one or more characters from their original password. However, due to the robust and extensible nature of Survey++, it can be used to test a wide variety of usability metrics relating to authentication processes or password generation and recovery.

Typically, researchers studying the usability and security of authentication and credential recovery mechanisms have relied on custom-written/home-grown platforms to test specific metrics that they are interested in. While these platforms are a good step towards collecting data, they lack portability and repeatability and make it difficult for other researchers to confirm or expand on initial findings. Survey++ fills this void as it aims to be a modular and extensible platform for usability testing in the realm of authentication and credential recovery. After collecting data with Survey++, a researcher may either export this data as secondary data to other researchers, or they may outline the configuration options that they used for their experiments so that other researchers using Survey++ can quickly and easily attempt to confirm their findings.

The rest of this short report on Survey++ is organised as follows: Section 2 gives background on authentication and access control. Section 3 gives a general overview of how Survey++ works and its major features. Section 4 discusses areas for future work involving extending Survey++. Finally, Section 5 concludes the report.

2 Background

Information security can be defined as “the preservation of confidentiality, integrity, and availability” [7]. To ensure confidentiality, systems usually depend on an authentication process that identifies a user as well as confirm that the user is who they say they are [1]. Confirming a user is who they say they are has been the purpose of 3 main authentication factors [6]:

- Knowledge factors: User is required to know a token, such as a password, passphrase, personal identification number, or response to a challenge.
- Ownership factors: User is required to possess an artefact such as a security token, ID card, hardware token or software token.
- Inherence factors: User authenticates using something they are or do. Examples include DNA, fingerprint, retinal pattern, signature, face, etc.

The typical authentication process has 3 facets:

1. Enrolment: A user and their corresponding authentication factor are added to the system.
2. Regular operation: The user is required to satisfy this authentication factor to login to the system for normal use.
3. Failure: In case a user fails to provide the correct authentication token, the system should provide a means of resetting this token.

Figure 1 shows a branching diagram which describes the way Survey++ handles these 3 facets. From the figure, we see that during enrolment, a token and recovery mechanism are assigned to a user. The researcher administering the experiment chooses whether the token will be user-generated or system-generated and also what type of password recovery policy will be used should a user forget their token. During regular operation, the user logs in to Survey++ normally, and answers survey questions. On a subsequent login to Survey++ (to complete a new part of a survey), in case the user forgets the token that was used previously, they will invoke the password recovery policy as specified by the administrator.

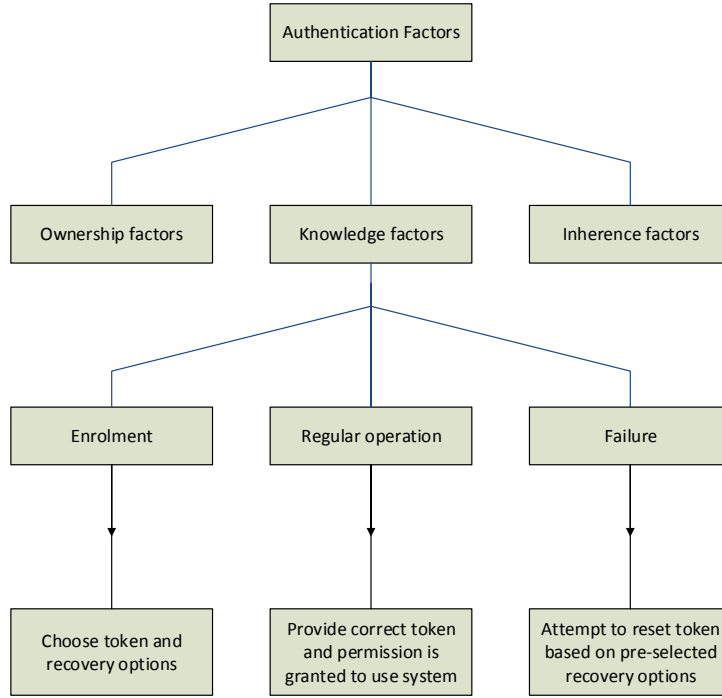


Fig. 1. Branching diagram showing various aspects of the authentication process used by Survey++.

Maxion argues that the hallmarks of a good experiment are repeatability, reproducibility and validity [5]. He argues that repeatability refers to an agreement in measurement (within measurement error) each time a measurement is taken. Reproducibility, according to Maxion, means that results of measurements should agree, irrespective of who takes the measurement. As for validity, Maxion states that validity is “the logical well-groundedness of how the experiment is conducted, as well as the extent to which the results will generalize to circumstances beyond those in the laboratory.” This argumentation solidifies the need for Survey++ as a platform for ensuring that usability experiments in the area of authentication and credential recovery meet the hallmarks which are expected of good and dependable experiments.

Survey++ ensures repeatability and reproducibility because the parameters for each experiment are selected by the researcher, and can be documented or exported to other researchers for the purpose of verifying the dependability of the experiment that was conducted. As for validity, Survey++ has a built-in model of a typical workflow for enrolment, regular operation, and credential recovery (recall Figure 1). This workflow approximates a common authentication paradigm in use today and contributes to the validity of the results obtained from using Survey++ as well as the generalisability of these results to authentication processes with a similar workflow. Extensions to this simplistic workflow model, which are described in Section 4, are designed to make experiments conducted using Survey++ dependable, even under emergent or novel workflows.

3 Inside Survey++

Survey++ is built to aid in the administering of experiments using knowledge factors for authentication, but its loosely-coupled design allows the researcher to add other authentication factors with minor effort, if necessary. Survey++ has 3 major loosely-coupled components that, when combined, delivered a modular and extensible usability testing platform. These 3 major components are:

- Survey module: The survey module serves as the distractor for the system and attracts the user’s attention away from the authentication and password recovery process to help ensure that no biases are introduced into the system. The administrator is able to add questions to the survey and decide how many sections the survey will be divided into. By having multi-section surveys, the researcher can measure the efficacy of their password recovery mechanism whenever a user comes back to complete a subsequent survey section and fails to remember their password.
The administrator chooses the window of time between sections of the survey and Survey++ automatically prompts users, via email, when they are required to complete a subsequent section. This configurable waiting time allows the researcher to have insights about recall accuracy versus time. The administrator can also choose from default survey questions or add their own; this allows them to, for example, collect demographic data on participants while they test these participants’ ability to recall their passwords.
- Authentication module: The authentication module handles enrolment, logins, and password recovery. For enrolment, this module allows the administrator to choose the password generation policy as well as relevant length/complexity/entropy characteristics.
The authentication module also handles the delivery of password hints. The administrator chooses the policy concerning the delivery of hints and this module applies the policy to the user’s password and generates their password hint whenever it is requested.
- Measurement module: The measurement module is responsible for logging the actions and timings of the various user interactions with the system. These measurements can then be analysed externally to obtain insights on user behaviour and password recall.

3.1 What it does

Here, we will briefly outline what happens on each step of the way when a usability experiment is being set up using Survey++.

The administrator will log into the administrative backend and choose a name for their survey. After this, Survey++ allows the administrator to add arbitrary or default questions to the survey. The default questions available to be added to surveys can be edited by editing the configuration file containing these default questions.

The administrator is then allowed to add users to the list of survey participants by supplying a list of email addresses. They then have the choice of how many sections the survey will have and the amount of time that should lapse after a user completes a section before Survey++ prompts the user to complete a subsequent section. These options are all variable and allow the researcher to fine-tune the experiment based on their specifications.

The final step of creating a survey requires the researcher to choose, from drop-down menus, the password generation policy and password reminder policy for the survey. Passwords may be user-generated or system-generated. For user-generated passwords, users are informed of the password requirements and then required to craft a password matching those requirements to use to login to the system. System-generated passwords can either be generated randomly or pulled randomly from a file containing a list of passwords. The inner workings and extensibility of the password generation policy and password reminder policy options are discussed in Section 3.2.

3.2 Design Considerations

Survey++ was built to allow a researcher to easily add their own password generation policies and password reminder policies in addition to those policies provided by default. For example, in Listing 1.1, we see sample XML code that, when added to the appropriate configuration file, causes an additional password policy to become available for the researcher to choose when creating a survey. The researcher only needs to specify password length and character set (and some other standard attributes) and Survey++ will randomly generate passwords to those specifications.

Listing 1.1. Sample XML for randomly generating passwords.

```
<option>
  <id>1</id>
  <title>Lower-case letters only, 4 chars. long</title>
  <source>random</source>
  <passwordlength>4</passwordlength>
  <characterset>abcdefghijklmnopqrstuvwxyz</characterset>
  <description></description>
</option>
```

Listing 1.2 shows XML code that is used to specify a password generation policy that selects a password randomly from a file containing a list of passwords.

Listing 1.2. Sample XML for selecting passwords randomly from a file containing passwords.

```
<option>
  <id>2</id>
  <title>Selected from file: passphrases.dic</title>
  <source>passphrases.dic</source>
  <passwordlength></passwordlength>
  <characterset></characterset>
  <description></description>
</option>
```

Listing 1.3 shows XML code that is used to specify a password generation policy that allows the user to choose their own password. In this case, the *characterset* tag contains a Perl-Compatible Regular Expression which describes to Survey++ what a valid user-chosen password would be.

Listing 1.3. Sample XML for allowing the user to choose their own password.

```
<option>
```

```

<id>3</id>
<title>Contains 6 or more lower-case letters</title>
<source>user</source>
<passwordlength></passwordlength>
<characterset>/^[a-z]{6,}$</characterset>
<description></description>
</option>

```

Finally, listing 1.4 shows XML code that is used to specify how password hints are given to a user. This function uses a mask system and incorporates asterisks to block out characters that should not be shown to a user during password hinting.

Listing 1.4. Sample XML for showing a password hint of the first character of password.

```

<option>
  <id>1</id>
  <title>Show 1st character of password</title>
  <mask>a*****</mask>
</option>

```

3.3 Lessons Learnt

While the development of Survey++ has been a success, some lessons have been learnt with regard to implementing a useful and extensible system for testing the usability of authentication and password recovery mechanisms. For example, while Survey++ is infinitely extensible in terms of password-generation and password-hinting policies, it suffers from problems with its static workflow model. For example, Survey++ assumes a workflow of enrolment, login, and potential password reset as discrete events. It cannot easily handle changes to that workflow, for example, multi-factor authentication or any additional steps such as vetting a user before giving a password hint. For this we perhaps need a language that specifies authentication events to the system and dictates valid stages of progression through a workflow. We discuss these and other directions for expansion in Section 4.

When administering surveys remotely, consideration will have to be given to those users that use in-browser password managers. These utilities would introduce errors into the data since they would make it seem as if users were better at remembering passwords than they actually are. We can use the Javascript timing features of Survey++ to help determine if password managers are in use and mitigate their corruption of good data. We can do this because password managers usually populate textfields instantaneously (faster than a human can type) and thus by recording the timings for text entry we could reject data from users, if necessary, if they are determined to be using password managers. In the case where usability studies are being conducted using Survey++ in a controlled environment, password managers will not be a problem since the researcher would have configured the test machines to their required specifications themselves.

4 Future Directions

Going forward, a number of challenges need to be addressed for Survey++ to remain useful. Sasse argues that the nature of authentication needs to be re-thought.

She advocates reducing the number of explicit authentication events by designing technologies that implicitly authenticate users [2]. Batt and Santhanam continue along these lines with a survey on keystroke dynamics for biometric authentication [3]. Survey++ needs to be able to handle various authentication factors and workflow models in a truly extensible way as was hinted at in Section 3.3.

A promising way forward that tackles both these challenges is the development of a formal scripting language for Survey++ that describes events and relationships between these events. In this way, the researcher would have the ultimate flexibility in specifying novel authentication and password recovery workflows which may not have been considered by the developers of Survey++. In addition to specifying workflows, the language could dictate to the underlying platform how to process various channels of input at each particular stage in the workflow. For example, we could script Survey++ so that it takes input from a gaze-tracker during enrolment and password recovery, but not during the normal use of the system.

Gaze-tracking is only the beginning. There is a plethora of other input, both biometric and non-biometric, that could be fed into Survey++ while users are completing their surveys. With a scripting language and processing engine built in to Survey++, we are able to overcome the limitations of its limited workflow model and at the same time cater to the needs of progressive researchers such as Sasse [2] and Batt and Santhanam [3].

Separate plugins could also be written for Survey++ that help to make sense of the data that is gathered. Assuming that various biometric and non-biometric data is being fed into the platform during usability testing, it becomes necessary to have a means of quickly filtering through this data to get the necessary insight that was sought. By building facilities into Survey++ that support the scripting of experiments and plugins to make sense of data, Survey++ would be able to satisfy the needs of various researchers while providing a means for the sharing of experiment parameters, data, and data processing.

4.1 What might a scripting language for Survey++ look like

A potential survey scripting engine for Survey++ could contain a GUI for making flowcharts which describe the workflows and decision making for an authentication process. In Figure 2 we show a flowchart that approximates the static workflow built into this version of Survey++.

The processes drawn in the flowchart (*workflow_enrolment*, *workflow_login*, *workflow_do_survey*, *workflow_failure*) could then be treated as functions with accompanying code scripted by the user in a novel Survey++ scripting language. This language could potentially contain features borrowed from access control languages such as XACML [4]. We envisage a scripting language containing keywords of various parts of speech such as:

- Nouns: *username*, *token*, *token_validity*
- Adjectives: *strong*, *weak*, *alphanumeric*
- Verbs: *generate*, *challenge*, *remind*, *revoke*, *assign*, *get*

In Listing 1.5 we see code in a theoretical scripting language that tells Survey++ how to handle the enrolment stage of the authentication process. The example code generates an alphanumeric token for a user and saves it in the database. The user is then shown the password and prompted to enter it. If successful, the user is allowed to do their survey.

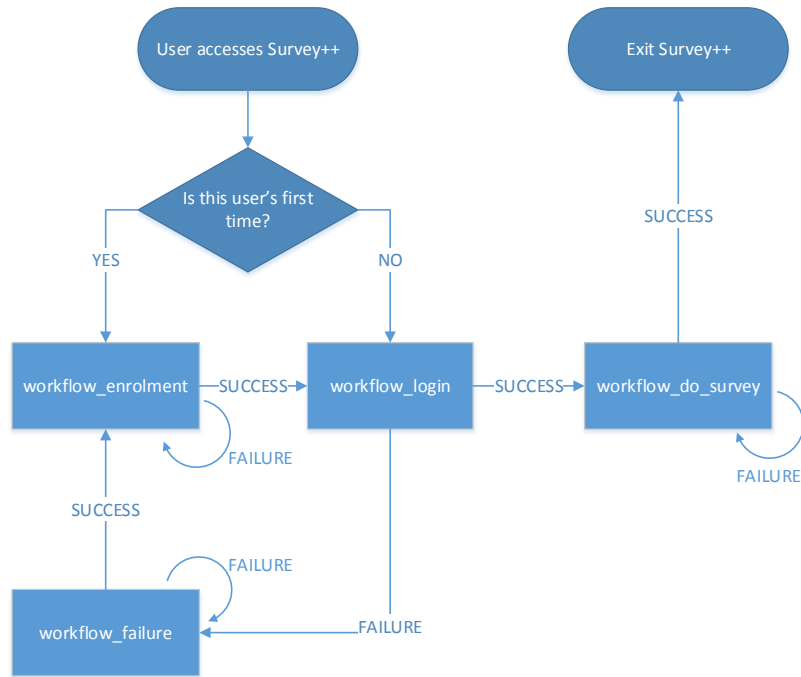


Fig. 2. Flowchart showing an approximation of the workflow in the current version of Survey++.

Listing 1.5. Theoretical script for enrolment process

```

function workflow_enrolment($username)
{
    RECORD user input;
    RECORD gaze-tracker input;
    GENERATE alphanumeric token => $generated_token;
    ASSIGN($username, $generated_token);
    SHOW $generated_token;

    // prompt user to enter newly generated password
    if (CHALLENGE())
    {
        workflow_login();
        workflow_do_survey();
    }
    else
    {
        workflow_enrolment();
    }
}

```

In Listing 1.6 we see code in a theoretical scripting language that tells Survey++ how to handle the login stage of the authentication process. The example code gets the username from the session cookie and then prompts the user to supply a token. The username and token are then verified and the appropriate action taken.

Listing 1.6. Theoretical script for login process

```
function workflow_login ()
{
    GET $_GLOBAL($username) => $username;
    CHALLENGE() => $supplied_token;

    if (VERIFY($username, $supplied_token))
    {
        workflow_do_survey ();
    }
    else
    {
        RECORD user input;
        workflow_login ();
    }
}
```

A scripting language like this would be a novel way for describing and generating experiments for testing various parameters of usability in both authentication and password recovery. This language would remove the limitations of the static workflow currently plaguing Survey++ in addition to making Survey++ almost infinitely extensible as a usability testing platform for authentication and credential recovery.

5 Conclusion

Survey++ is a useful tool for performing certain types of usability experiments in the realm of credential recovery and authentication. Although very extensible in some regards, Survey++ fails to be very extensible when handling novel workflows for both authentication and password recovery. A promising avenue to solve these problems is to develop a scripting language in which usability experiments can be written. This scripting language would describe workflows, password generation and reminder policies, and multiplexing/demultiplexing sensors capturing user behaviour and input during testing. After interpreting these scripts, the platform could automatically generate and launch a survey with the prescribed specifications. As computer systems become an integral part of our lives, and as long as the need for authenticating users remains, we will need to find new ways to satisfy security requirements while not unnecessarily burdening a user during authentication and credential recovery.

References

1. Adams, A., Sasse, M.A.: Users are not the enemy. *Commun. ACM* 42(12), 40–46 (Dec 1999), <http://doi.acm.org/10.1145/322796.322806>

2. Angela Sasse, M.: Technology Should Be Smarter Than This!: A Vision for Overcoming the Great Authentication Fatigue. In: Jonker, W., Petkovi, M. (eds.) *Secure Data Management*, pp. 33–36. *Lecture Notes in Computer Science*, Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-06811-4_7
3. Bhatt, S., Santhanam, T.: Keystroke dynamics for biometric authentication - a survey. In: *Pattern Recognition, Informatics and Mobile Engineering (PRIME)*, 2013 International Conference on. pp. 17–23 (Feb 2013)
4. Godik, S., Moses, T., Anderson, A., Parducci, B., Adams, C., Flinn, D., Brose, G., Lockhart, H., Beznosov, K., Kudo, M., et al.: extensible access control markup language (xacml) version 1.0 (2003)
5. Maxion, R.: Making experiments dependable. In: Jones, C., Lloyd, J. (eds.) *Dependable and Historic Computing*, *Lecture Notes in Computer Science*, vol. 6875, pp. 344–357. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-24541-1_26
6. O’Gorman, L.: Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE* 91(12), 2021–2040 (Dec 2003)
7. Parker, D.B.: Restating the foundation of information security. *Computer Audit Update* 1991(10), 2–15 (1991)