Excerpts from
Initial Research: Selected E-Docs Providers

Organizational details withheld

Gary Arthur Douglas II

Excerpts from

Initial Research: Selected E-Docs Providers

Organizational details withheld

Gary Arthur Douglas II

TABLE OF CONTENTS

T

ABLE OF

C

ONTENTS

Summary

In support of efforts to select "Electronic Document" ("E-Docs") vendors for the company, WidgetSec has performed a brief informal analysis of three candidate products: Recombo, DocuSign and GIS.

This does not represent the results of a complete, formal security assessment.

Briefly reviewed was product documentation, exposed source code and application logic, remote services and site security for each of the three vendors.

A simplified analysis is then provided, intended to help guide the vendor selection process.

Analysis

In general, two of three vendors (DocuSign and GIS) are considered comparable, as regards security, in terms of design and execution. DocuSign, in particular, is known to WidgetSec (selected notes from earlier research is included herein).

A third (Recombo), was found seriously deficient, leading the author to consider it unsuitable for the intended purpose. Deficiencies include flaws in architecture as well as implementation which likely expose the application's users to serious risk of data breach, manipulation, and regulatory liability.

While a formal analysis remains appropriate prior to engaging either DocuSign or GIS on a formal basis, this report concludes that no such effort is required the case of Recombo.

The remainder of this document includes selected notes from this research effort.

Summary In support of efforts to select "Electronic Document" ("E-Docs") vendors for the

company, WidgetSec has performed a brief informal analysis of three candidate products: Recombo, DocuSign and GIS.

This does not represent the results of a complete, formal security assessment.

Briefly reviewed was product documentation, exposed source code and application logic, remote services and site security for each of the three vendors.

A simplified analysis is then provided, intended to help guide the vendor selection process.

Analysis In general, two of three vendors (DocuSign and GIS) are considered comparable, as regards security, in terms of design and execution. DocuSign, in particular, is known to WidgetSec (selected notes from earlier research is included herein).

A third (Recombo), was found seriously deficient, leading the author to consider it unsuitable for the intended purpose. Deficiencies include flaws in architecture as well as implementation which likely expose the application's users to serious risk of data breach, manipulation, and regulatory liability.

While a formal analysis remains appropriate prior to engaging either DocuSign or GIS on a formal basis, this report concludes that no such effort is required the case of Recombo.

The remainder of this document includes selected notes from this research effort.

Research Notes

Recombo

The product produced by Recombo (recombo.com) is called "Agreement Express." The Recombo.com web site itself does not control access to the application, which is hosted from agreementexpress.com. However, serious issues with recombo.com strongly suggest a weak commitment to security – an implication further supported by this informal analysis.

In dramatic contrast to industry-standard practice, the "Agreement Express" application appears to be comprised of a handful of "monolithic" JavaScript files, and Adobe Flex – most of which are externally accessible to unauthenticated parties. This model exposes the site and its users to an immeasurable range and variety risks.

Noteworthy items from this brief examination included the following.

- The recombo.com web site is Wordpress – with the admin UI exposed (http://www.recombo.com/wp-login.php)
- Source code for end-user login processing (and application) reflects a design philosophy typified by, among other weaknesses, poor user input validation implemented in client-side code - an example - http://us.agreementexpress.net/flex/login.jsp
- The main application engine (and logic) is exposed to unauthenticated users via the "Manager" application code: https://us.agreementexpress.net/wp20/wpmanager/manager/manager.jsp
- Note that there are separate US, and "CA/EU" logins. This warrants closer investigation.
- "Manager" application code demonstrates a host of serious security issues including:
    - Poor user input validation, implemented in client-side code
    - Weak application logic which may be susceptible to manipulation
    - Some transactions processing functions can be accessed *prior to authentication*
- "Manager" application code is full of transitory and "hack/mod" code. For instance, library samples included (though disused), informative legacy code is commented out (some of it very worrying).
- "Manager" application is likely susceptible to wide variety of manipulation. Various malformed and incomplete transactions have produced sometimes informative results.
- Recombo API: Agreement Express "integrates with your website." We must determine how this is implemented, and secured.
- Digital I-9 processing: The product optionally processes I-9 documents with DHS. *The business logic and communications flow* of this process is exposed in the "Manager" application! Administrators have access sufficient to review I-9s. This is very worrisome.
- According to the Recombo FAQ, **"you will be emailed your password..."**, and this author does not doubt it. http://www.recombo.com/support/faq/my-password/

    - Bonus: The same article tells us of the max length, and weak charset for user passwords – know we know the keyspace. Interestingly, passwords may not start with a number. That is quite odd, and possibly informative. The password change process is validated in client-side code, such that malicious users could set any password they like, unless there is additional back-end validation – which seems unlikely given the observed design "methodology."

Research Notes

Recombo The product produced by Recombo (recombo.com) is called “Agreement Express.” The Recombo.com web site itself does not control access to the application, which is hosted from agreementexpress.com. However, serious issues with recombo.com strongly suggest a weak commitment to security – an implication further supported by this informal analysis.

In dramatic contrast to industry-standard practice, the “Agreement Express” application appears to be comprised of a handful of “monolithic” JavaScript files, and Adobe Flex – most of which are externally accessible to unauthenticated parties. This model exposes the site and its users to an immeasurable range and variety risks.

Noteworthy items from this brief examination included the following.

• The recombo.com web site is Wordpress – with the admin UI exposed (http://www.recombo.com/wp-login.php)

• Source code for end-user login processing (and application) reflects a design philosophy typified by, among other weaknesses, poor user input validation implemented in client-side code - an example - http://us.agreementexpress.net/flex/login.jsp

• The main application engine (and logic) is exposed to unauthenticated users via the “Manager” application code: https://us.agreementexpress.net/wp20/wpmanager/manager/manager.jsp

• Note that there are separate US, and “CA/EU” logins. This warrants closer investigation.

• “Manager” application code demonstrates a host of serious security issues including:

o Poor user input validation, implemented in client-side code o Weak application logic which may be susceptible to manipulation o Some transactions processing functions can be accessed prior to authentication

• “Manager” application code is full of transitory and “hack/mod” code. For instance, library samples included (though disused), informative legacy code is commented out (some of it very worrying).

• “Manager” application is likely susceptible to wide variety of manipulation. Various malformed and incomplete transactions have produced sometimes informative results.

• Recombo API: Agreement Express “integrates with your website.” We must determine how this is implemented, and secured.

• Digital I-9 processing: The product optionally processes I-9 documents with DHS. The business logic and communications flow of this process is exposed in the “Manager” application! Administrators have access sufficient to review I-9s. This is very worrisome.

• According to the Recombo FAQ, "you will be emailed your password...", and this author does not doubt it. http://www.recombo.com/support/faq/my-password/

o Bonus: The same article tells us of the max length, and weak charset for user passwords – know we know the keyspace. Interestingly, passwords may not start with a number. That is quite odd, and possibly informative. The password change process is validated in client-side code,

such that malicious users could set any password they like, unless there is additional back-end validation – which seems unlikely given the observed design “methodology.”

Regarding "Manager.jsp"

The *source code is very very illuminating*! https://us.agreementexpress.net/wp20/wpmanager/manager/manager.jsp

The changePassword() function may be the easiest way to brute user accounts, and change passwords, at one swipe: There will of course be other hoops to clear first… An extract:

```
var sUrl = /ChangePassword?login="+emailAddress+
    "&password="+oldPassword+
    "&newPassword="+newPassword2;
```

The logindialog.js (like the rest of this source) is sloppy and loose, and appears weak to *every* form of manipulation – we can only hope it's role in logging in is not as it appears.

https://us.agreementexpress.net/wp20/wpmanager/manager/js/logindialog.js

```
function onLoginDialogSubmit(){
    var validationResult=validateLoginParameters();
    if (validationResult===""){
    var  oldErrMessage=
            trimAll(document.getElementById("dlgLoginFormLoginError").innerHTML);
      if (oldErrMessage !== "" && oldErrMessage!=" "){
            document.getElementById("dlgLoginFormLoginError").innerHTML=" ";
      }
    var sUrl = "/Login?login="
        +document.getElementById("dlgLoginFormLogin").value
        +"&password="+document.getElementById("dlgLoginFormPassword").value
        +"&contractLogin="+document.getElementById("contractLogin").value
        +"&launchURL="+document.getElementById("launchURL").value
        +"&loginType=1";
    var request = YAHOO.util.Connect.asyncRequest('POST', sUrl, callbackLogin);
    (…)

function sendPassword() {
  if (validateNotEmpty(document.getElementById("dlgLoginFormLogin").value)==false){
    Ext.Msg.alert('Error', 'Please enter your Email Address');
        } else {
            getSecurityQuestion(document.getElementById("dlgLoginFormLogin").value);
        }
}
```

https://us.agreementexpress.net/wp20/wpmanager/manager/manager.jsp

The changePassword() function may be the easiest way to brute user accounts, and change passwords, at one swipe: There will of course be other hoops to clear first... An extract:

•

var sUrl = /ChangePassword?login="+emailAddress+

"&password="+oldPassword+ "&newPassword="+newPassword2;

The logindialog.js (like the rest of this source) is sloppy and loose, and appears weak to every form of manipulation – we can only hope it's role in logging in is not as it appears.

https://us.agreementexpress.net/wp20/wpmanager/manager/js/logindialog.js

function onLoginDialogSubmit(){

var validationResult=validateLoginParameters(); if (validationResult===""){ var oldErrMessage=

trimAll(document.getElementById("dlgLoginFormLoginError").innerHTML); if (oldErrMessage !== "" && oldErrMessage!=" "){

document.getElementById("dlgLoginFormLoginError").innerHTML=" "; } var sUrl = "/Login?login="

+document.getElementById("dlgLoginFormLogin").value
+"&password="+document.getElementById("dlgLoginFormPassword").value
+"&contractLogin="+document.getElementById("contractLogin").value
+"&launchURL="+document.getElementById("launchURL").value +"&loginType=1"; var request = YAHOO.util.Connect.asyncRequest('POST', sUrl, callbackLogin); (...)

function sendPassword() {

if (validateNotEmpty(document.getElementById("dlgLoginFormLogin").value)==false){

Ext.Msg.alert('Error', 'Please enter your Email Address');

} else {

getSecurityQuestion(document.getElementById("dlgLoginFormLogin").value); } }

Email addresses discovered at Recombo.com
Located in documents or source code, then used to test login mechanisms (see later).

- balvarez@recombo.com
- ecorreia@agreementexpress.co.uk
- ijoffe@recombo.com
- maizer@recombo.com
- mgardner@recombo.com
- recombo_sales@recombo.com
- recombo_support@recombo .com
- support@recombo.com

Email addresses discovered at Recombo.com Located in documents or source code, then used

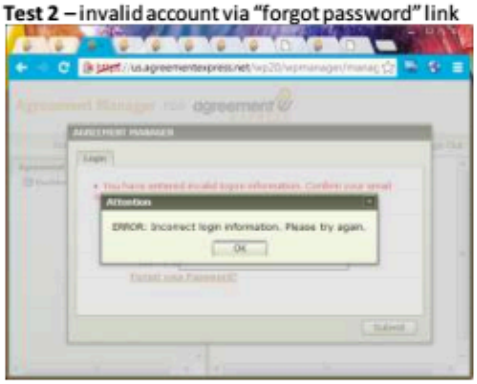
to test login mechanisms (see later).

- balvarez@recombo.com
- ecorreia@agreementexpress.co.uk
- ijoffe@recombo.com
- maizer@recombo.com
- mgardner@recombo.com
- recombo_sales@recombo.com
- recombo_support@recombo .com
- support@recombo.com

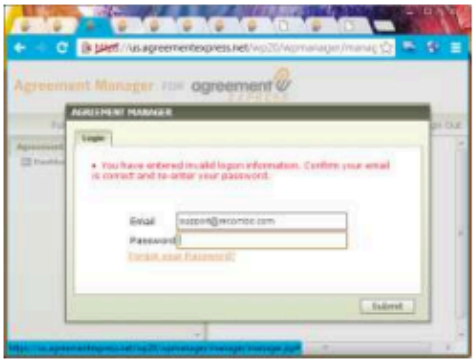
Recombo: Account Iteration (and probably lots more) via Login UI / Partial Use of "Recovery" Mechanism

Test 1 – invalid (assumed) account (via 'submit')

Test 2 – invalid account via "forgot password" link



Test 3 – Guessed (!) account validated (silently) via "Forgot Password" – bonus, we get their personal security question! This appears to be some Ajax call – so (bonus!) we could probably script a simple user account enumeration process. *Tasty*.



More fun with this functionality! Including:

support@recombo.com
test@recombo.com
balvarez@recombo.com
recombo_support@recombo.com

“Recovery” Mechanism

Test 1 – invalid (assumed) account (via ‘submit’) Test 2 – invalid account via “forgot password” link

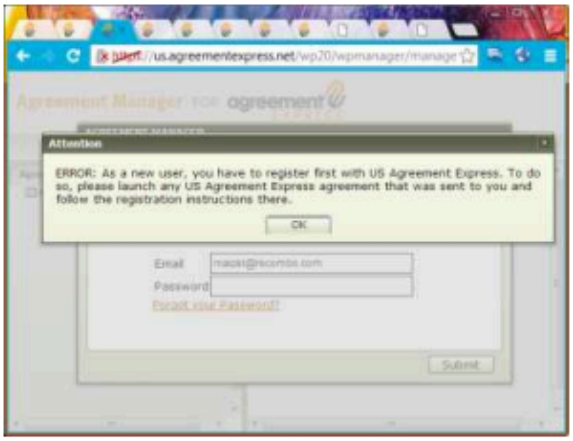
Test 3 – Guessed (!) account validated (silently) via “Forgot Password” – bonus, we get their personal security question! This appears to be some Ajax call – so (bonus!) we could probably script a simple user account enumeration process. Tasty.

More fun with this functionality! Including:

support@recombo.com test@recombo.com balvarez@recombo.com recombo_support@recombo.com

More fun with Recombo.com “manager.jsp” recovery feature...

Different response - this appears to be a brand new account?? “maizer@recombo.com”



Different response - this appears to be a brand new account?? “maizer@recombo.com”

DocuSign

Brief

A previous brief analysis of DocuSign exposed (and/or suggested) various deficiencies in operational methods and procedures, but was generally favorable regarding the application itself.

However, as no formal application security assessment was undertaken, it would be presumptuous to declare the product itself safe for the company at this time.

Any such analysis should take care to examine the following, which have been considered of relevance:

- June 2012 "Disclosure" (not breach) incident: Links to signed documents exposed via Google/other search engines. Internal server/site misconfiguration was almost certainly the cause. http://agbeat.com/tech-news/docusign-security-breach-users-private-info-leaked-to-web-breaking/http://agbeat.com/tech-news/docusign-security-breach-users-private-info-leaked-to-web-breaking/
- Some server-side features make use of "PDFKit.net", a potentially dangerous component. Their version appears to be out-of-date. Some anti-virus engines even mistake this for malware.
- Discussions with DocuSign technical teams have assured this author of the suitability of internal security processes and architecture.
- Extensive DocuSign API warrants thorough analysis
- DocuSign has ISO 27001 certification, others – and attestations.

Brief

A previous brief analysis of DocuSign exposed (and/or suggested) various deficiencies in operational methods and procedures, but was generally favorable regarding the application itself.

However, as no formal application security assessment was undertaken, it would be presumptuous to declare the product itself safe for the company at this time.

Any such analysis should take care to examine the following, which have been considered of relevance:

• June 2012 “Disclosure” (not breach) incident: Links to signed documents exposed via Google/other search engines. Internal server/site misconfiguration was almost certainly the cause. http://agbeat.com/tech-news/docusign-security-breach-users-private-info-leaked-to-web-breaking/http://agbeat.com/tech- news/docusign-security-breach-users-private-info-leaked-to-web-breaking/

• Some server-side features make use of “PDFKit.net”, a potentially dangerous component. Their version appears to be out-of-date. Some anti-virus engines even mistake this for malware.

• Discussions with DocuSign technical teams have assured this author of the suitability of internal security processes and architecture.

• Extensive DocuSign API warrants thorough analysis

• DocuSign has ISO 27001 certification, others – and attestations.

GIS ("Pangea")

Brief

Very little analysis has been performed on Pangea, and very little is possible without some level of access. Initial reconnaissance suggests a highly-secured application front-end with very little "low-hanging fruit" apparent at first glance. There is, for instance, no visibility at all into application logic exposed to unauthenticated parties (in contrast to Recombo).

The industry holds Pangea (and GIS) in high esteem, and they seem to have a complete suite of certifications.

- The product documentation does not imply the availability of a safe authentication capability. This must be investigated.
- Pangea features an API – this too must of course be vetted. How are operations authenticated, authorized, tracked, audited, etc?

GIS ("Pangea")

Brief

Very little analysis has been performed on Pangea, and very little is possible without some level of access. Initial reconnaissance suggests a highly-secured application front-end with very little “low-hanging fruit” apparent at first glance. There is, for instance, no visibility at all into application logic exposed to unauthenticated parties (in co ntrast to Recombo).

The industry holds Pangea (and GIS) in high esteem, and they seem to have a complete suite of certifications.

• The product documentation does not imply the availability of a safe authentication capability. This must be investigated.

• Pangea features an API – this too must of course be vetted. How are operations authenticated, authorized, tracked, audited, etc?