Did You Ever Have To Make Up Your Mind? What Notes Users Do When Faced With A Security Decision

Mary Ellen Zurko, *IBM Software Group*, *mzurko@us.ibm.com*
Charlie Kaufman, *IBM Software Group*
Katherine Spanbauer, *IBM Software Group*
Chuck Bassett, *IBM Software Group*

Abstract

Designers are often faced with difficult tradeoffs between easing the user's burden by making security decisions for them and offering features that ensure that users can make the security decisions that are right for them and their environment. Users often do not understand enough about the impact of a security decision to make an informed choice. We report on the experience in a 500-person organization on the security of each user's Lotus Notes client against unsigned active content. We found that the default configuration of the majority of users did not allow unsigned active content to run. However, we found that when presented with a choice during their work flow, many of those otherwise secured users would allow unsigned active content to run. We discuss the features that are in Lotus Notes that provide security for active content and that respond to the usability issues from this study.

1. Introduction

Applications writers and software developers are often faced with design decisions about what security mechanisms to provide, how those mechanisms are presented to the various types of users, and what policies are possible, are privileged, and are shipped by default with those mechanisms. Powerful new features can often present a tradeoff between staying out of the user's way while they do their work and providing minimal protection if the features are used maliciously, or making sure that nothing potentially dangerous happens unless someone on site (user, administrator) has determined that the specific situation poses an acceptable risk. Features are often shipped with settings that allow for either choice, but the default settings still need to be chosen. For example, in February 2002, Microsoft spent the month on security related activities that included changing defaults in security-related options from open to secured [2].

It's hard to figure out if something is safe or not. For example, it is convenient for an email message to include buttons so the recipient can RSVP with a 'yes' or 'no' response, but a fully general capability to automatically generate and send email can be used to spread viruses and compromise the recipient's privacy. The more general purpose a feature is, the harder it is to determine whether it is safe. In our example case of sending an email message, the system can compare the source of the original email message with the destination of any email it tries to send, and whether the contents of the email is hard coded or dynamically generated, as inputs to the decision process, but there will always be a gray area. If the mail message is trying to send a message back to its original sender (and both the address of the original sender and the potential recipient are verifiable), if the contents of that message is hard coded, and if the active content in the button has not accessed any information outside of the originally sent message, then it is probably safe. Other variations might also be safe. If the principal is both the current user and the signed code signer, the operation is probably trustworthy. Dynamically composed mail messages may be considered safe if they are displayed to the user before they are sent, but there can be problems with that as well, such as white on white font, information that doesn't make sense (bit level information, encrypted information), and hidden coded information (steganography).

If the system detects something dangerous, it has a number of choices. It can ignore the danger and proceed, it can disallow the action silently, it can disallow the action and report the failure either to a user or a system administrator (perhaps via a log file), or it can ask the user what to do. If the application designer can't write code that figures out whether an action is proper or improper, he has to leave open multiple of these options and pass the buck to the system administrator (who might know, for example, which sources of instructions should be trusted). If the system administrator can't figure out what's proper and manipulate configuration settings to reflect that

judgement, he has to either leave the system unsafe or dysfunctional, or leave the decision to the user. If the default setting is unsafe, the system administrator is likely to leave it that way because it generates no complaints. It would be highly unusual for the user to have enough information to determine whether the action is proper or not. The warning box might as well say "Do you feel lucky?". Users are likely to feel that if they are being given an option, then the application designer and system administrator must have already concluded that it is safe to proceed. Especially if they are being asked - in effect - whether they want to do their job unsafely or not do it at all.

Within that context, this paper discusses the results of a user study conducted just after the default security setting on an active content protection mechanism was changed from “open” to “secured.” The next section discusses that feature and the short-term impact on development when the default was changed. Section 3 covers how the user study was run. Section 4 describes the results of the study, and discusses the implications and reactions. Section 5 outlines changes in the next release related to the problems uncovered as a result of securing the defaults, and those uncovered as a result of running the user study itself. Section 6 covers related work, and the final section summarizes all of the conclusions in this paper and touches on potential future work in this area.

2. Lotus Notes Execution Control Lists

Lotus Notes is a platform for distributed applications, of which email and discussion forums are examples. As with web browsers, there is an expectation that users can visit and interact with sites (or databases in Notes parlance) safely, meaning that even if those sites are mismanaged they cannot corrupt the user's private data or interactions with other sites. It is useful, however, to be able to use this mechanism to do unsafe operations, like upgrading software on the workstation or composing and sending email messages based on filling out a form. These same techniques, however, can be used to steal user data or spread viruses if the application is not trustworthy. With both Lotus Notes and browsers, this is done with a combination of local configuration deciding certain operations should proceed without warnings and pop up dialogs asking users whether to override the defaults. Lotus Notes takes advantage of its built in public key infrastructure to digitally sign potentially dangerous active content so that trust decisions can be based on who created it rather than what server it is fetched from. This concept is particularly powerful in the context of email, where all of the email comes from the user's mailbox, but trust of its contents is based on the signature of the sender.

Execution Controls Lists (ECLs) were introduced in Lotus Notes version R4.5 to provide client-side security controls for the execution of potentially dangerous active content in the languages supported by the Notes client: LotusScript and @ formulas, Java, and Javascript. Active content in any of those forms can be embedded in either the design of a database and its documents (the forms), or in a document instance itself. A Notes signature is attached to the embedded active content, consisting of a hash of the active content, the hash signed with the private key of the person embedding it, and the certificate(s) associating the private key's corresponding public key with the name of the signer.

Each Notes client user has a Execution Control List (ECL). For each of the three active content languages, it lists the signers to which the user grants active content permissions, and the permissions granted. For example, in Figure 1, LotusScript or @ formulas signed by Jane Done/SoftwareHouse can read and write information in the database in which the active content is running and access environment variables. Java and Javascript execution controls are set and viewed on other panes of the ECL dialog box. Permissions can be added or removed from any signing identity, even the ones that are shipped with permissions, such as Lotus Notes Template Development/Lotus Notes.

A new user's ECL is initialized in two places. It includes the entries from the Administrative ECL set on the user’s home server, which specifies the signers that are trusted by the site administration. It also includes the default permissions required by Lotus Notes and Lotus companion products:

- The "-No Signature-" entry specifies the permissions given to unsigned active content.
- The "-Default-" entry specifies permissions given to active content with a signature that can be verified when the signer does not appear elsewhere on the user's ECL.

These two entries are the only ones that cannot be removed from the ECL. The permissions for any entry can be modified (enabled or disabled).

The Refresh button on a user's ECL updates that ECL with the current information from the administrative ECL. Each entry in the administrative ECL is added to the user's ECL, or overwrites the existing entry on the user's ECL if an entry with the same name already exists.

When active content running in the Notes client attempts a protected operation that requires an ECL permission, such as creating a new document in the current database, a check for the appropriate permission is made. If the signer has the permission, the active content continues running. If it doesn't, the user is prompted with an Execution Security Alert (ESA) indicating the action the code is attempting, who signed it, and the permission is required to complete the operation.

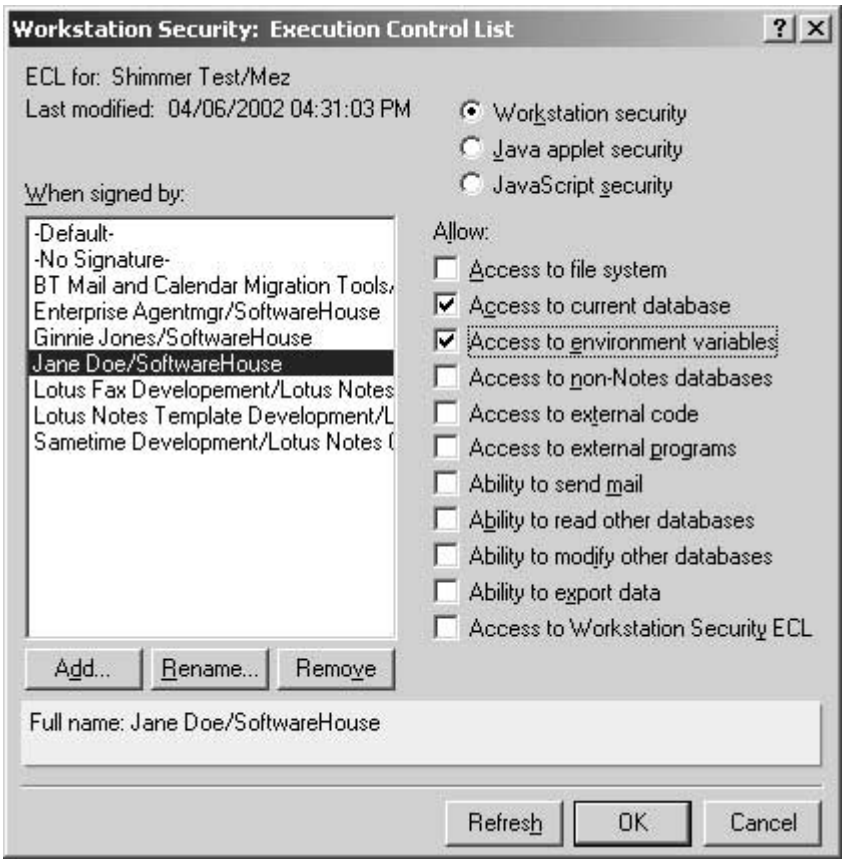

Figure 1: Lotus Notes Execution Control List

In Figure 2, a piece of active content signed by Lotus Notes Template Development/Lotus Notes (according to a public key certificate signed by the Notes CA) is calling the GetProfileField command to get a field from a profile document in the current database. That operation requires the "Access to current database" permission. The user can chose to abort that call (the default), execute the call this one time, or give the required permission to the signer in their ECL. The exception is those users who are not allowed to modify their own ECL (an option specified in the Administrative ECL) can only choose Abort.

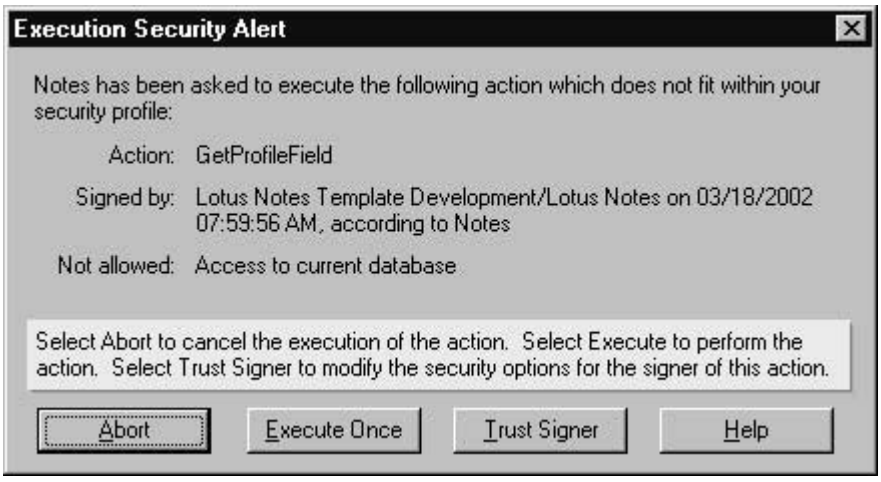

Figure 2: Execution Security Alert (ESA)

In R5.02 IBM began shipping tight defaults on ECLs. Originally, ECLs were initialized with all permissions given to the -No Signature- and -Default- entries. Sites could tighten that default for their users by removing permissions on those entries in the Administrative ECL. Not many did. Sites who tried to deploy tight ECL defaults found problems -- both in the features available for deployment and in some bugs. Since the initial defaults on ECLs were open, new features in product development were tested with them open. Some new features introduced bugs that would only be noticed when unsigned code was not allowed to execute. Committing to ship with secure ECL defaults meant a commitment to rapidly fix the bugs that appeared as a result of those defaults.

3. The User Study

With the change to shipping secure ECL defaults, any new users would get them automatically (unless their site administrator overrode them). The propagation of these defaults to existing users with the R5 features was more problematic. We were particularly concerned that as many users as possible begin running with no permissions given to unsigned active content as soon as they received the software update. We ran a pilot study at a small company that does Notes-related development (which we will call SoftwareHouse) to find out if tight ECLs could be deployed properly with R5.02 features.

First, the site Administrative ECL defaults were secured. Then, a prestigious SoftwareHouse security maven sent an email announcement to a list that included everyone in the company. It included a button for recipients to click in order to tighten their ECLs. The action of the button caused the the new administrative defaults to be merged into the user's ECL. The announcement included an extensive explanation of how the security of in-house ECLs was being tightened. It also provided an example of an unsigned Execution Security Alert, explaining the potential danger it could represent, and gave instructions about how to handle it safely and to whom to report it.

Three months later, we used active content in a survey email to obtain data on the state of ECLs in SoftwareHouse. The active content is shown in Figure 3., and was associated with the Postopen event of the design (form) of the survey mail message. It creates a mail message (document) to send. The creation of the document (Dim doc As New notesdocument (db)) calls a protected operation, since it accesses the current database. The mail message is then set to be sent back to the person conducting the survey (Jane Doe/SoftwareHouse). The actual sending of the document (doc.send) calls a second protected operation.

```
Sub Postopen(Source As Notesuidocument)
     On Error Resume Next
     Dim current As String
     current$ = Time$()
     Dim sess As New notessession
     Dim db As notesdatabase
     Set db = sess.currentdatabase
     Dim doc As New notesdocument(db)
     doc.SendTo="Jane Doe/SoftwareHouse"
     doc.Subject = "I allow unsigned code to execute on my workstation " + current + "; "
+ Time$()
     Call doc.send(False)
End Sub
```

Figure 3: Survey LotusScript Active Content

The mail message has two timestamps in the subject line. The first is the time taken before the first protected operation is called. The second time stamp is after that. If the call to the protected operation does not generate an alert, the difference in the timestamps will be minimal (0 - 1 seconds). If the call does generate an alert, it will require user interaction, and the difference in the timestamps will be 2 seconds or more. If the mail message is sent, and the timestamps are 2 or more seconds apart, the user must have chosen "Execute Once" or "Trust Signer" for both alerts generated by the code. The timestamps are also an indication of how much thought the user gave to the question. Figure 4 shows the second alert.

In order to attach active content to the design of a document, the document had to be sent with its own design (form), as opposed to using one in the mail database to which it was going. The form and its active content were not signed. Therefore, the operations requiring ECL permissions would be checked against the -No Signature- entry in the reader's ECL.

Users saw new mail from their colleague Jane Doe/SoftwareHouse, with the subject "Important Information". When they opened the email (or when they selected it with the preview pane that displays message content enabled), if their ECLs did not allow unsigned code the ability to access another database, a "-No Signature-" Execution Security Alert came up, covering up much of the content of the mail message. Since the mail message used a vanilla stored form, the top quarter of the displayed message was missing the Note mail look and

feel of a formatted header section with the sender's name on the left hand side and the To:, CC:, BCC: and Subject: fields on the right hand side, in fields. They could move that dialog, or respond via one of the buttons. If they glanced down at the bottom edge of the Notes client, they would have seen the status message "Signed by Jane Doe/SoftwareHouse on 11/22/1999 02:20:46 PM, according to /SoftwareHouse". If they responded with "Execute Once" or "Trust Signer", and if their ECL did not allow unsigned code to send mail on their behalf, the second ESA would appear (Figure 4). Dismissing that dialog in any way finished uncovering the content of the mail message.

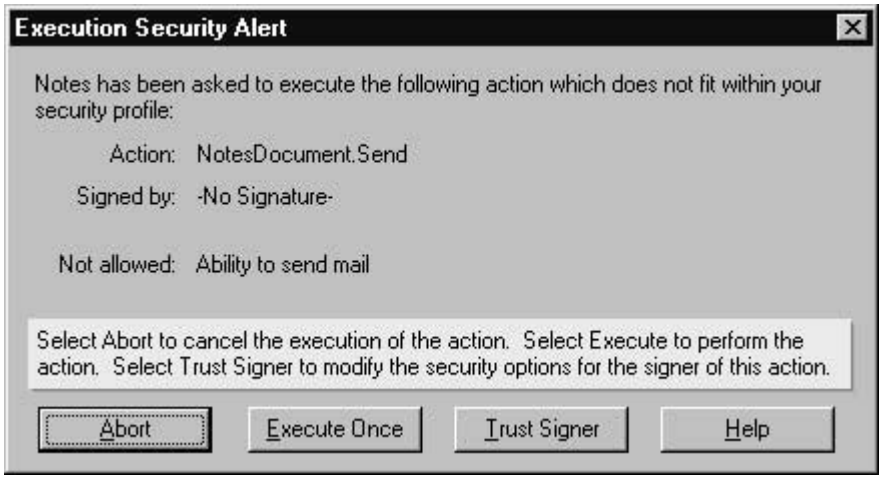

Figure 4: The alert resulting from sending mail from unsigned LotusScript

The text of the mail survey is shown in Figure 5. It explained the survey and encouraged people who didn't allow the unsigned code to execute to send the contact person mail telling them so, for data gathering purposes (the Gold Star list). The person who sent the survey was a SoftwareHouse security person more junior than the one who had sent the original email with the instructions on securing ECLs. It went to the same list that had been used to send the earlier mail message.

Subject: Important Information

Dear colleague,

I'm Jane Doe, from the SoftwareHouse security group. In an effort to determine how useful and effective our efforts have been in asking all of you to tighten up your workstation ECLs, we are collecting data on whether anyone here at SoftwareHouse still allows unsigned code to automatically execute on their workstation (and whether anyone who doesn't would still allow unsigned code of dubious origin to execute).

If your workstation ECL is still wide open, you did not see any alerts when you viewed this message, and you have sent me email telling me that you allow unsigned code to execute on your workstation. If you did see the alerts, and you allowed unsigned code to execute by pressing the "Execute Once" or "Trust Signer" buttons on those two Execution Security Alerts, you also sent me that message. If you routinely let hostile code execute in this fashion, the consequences could have been much worse (such as erasing your hard drive or leaking Product Y development secrets).

If you saw the alerts and aborted the code execution, thank you. You can further help us by sending me (Jane Doe) email telling me that you did that, allowing me to add you to our Gold Star list of colleagues who practice good ECL hygiene.

Thank you for participating in our study. Feel free to send me any questions or concerns.

Jane Doe

Figure 5: Text of survey message

4. Results and Discussion

4.1. Core Results

Figure 6 (below) shows the results from the first two days of responses to the survey. There were 543 names on the SoftwareHouse email list (after discounting the most obvious duplicate names). The "Other" category includes bounced deliveries and Out of Office notices. Responses dribbled in for up to two months after the mail was sent; they are not included in our discussions of the results.

62% of the people on the email list (334) responded within two days of the survey. 38% (209) of the potential recipients didn't respond within those two days. They may have not read the mail, they may have aborted the active content but not self-reported as Gold Stares, or they may have been MIME users. (LotusScript is stripped out of email sent to people configured to receive MIME). Three respondents self-reported on this, and there were 10 users in the SoftwareHouse domain configured this way.

68% of the respondents (227) – or 42% of the survey population -- did not allow unsigned code to access another database and send mail on their behalf. 31% of the respondents (102) did have open ECL defaults. A minimum of 42% (% of Total Sent with Secure ECL defaults) and a maximum of 82% (all but the 18% with verifiable Open ECL defaults) of the survey population had secure ECL defaults. Note that having secure ECL defaults does not ensure secure operation (as we discuss below). For a security configuration that must be explicitly set by users, the circumstances were close to ”as good as possible.” A mail message with a button to press had been sent out previously, and the user population had familiarity with the technology involved. 28% of the respondents (92) did not execute the unsigned code on their workstation. 56% of the target population did not execute the unsigned active content (non-response, Gold Star, and other). This still leaves a window of 44% of the target population that did execute unsigned active content, including 18% of the overall population that does so automatically. 44% of a target population is a large-enough security hole to do considerable damage. This is substantially larger than the 18% of the target population that did not even have a base configuration disallowing the dangerous behavior (unsigned active content) by default. This indicates that security mechanisms that can disallow dangerous behavior based on configuration, without a user override, have a much greater chance of doing so. Whether they can do so and still provide competitive and useful features is the challenge.

		% of Total Sent	% of Responses	% of Secure defaults
Names on list	543			
Responses	334	62%		
Open ECL defaults	102	18%	31%	
Secure ECL defaults	227	42%	68%	
Gold Star	92	17%	28%	41%
Clicked and sent	135	25%	40%	59%
Other	5	1%	1%	

Figure 6: Data from responses to email survey

Of all the respondents we know of that saw the ESA (Secure ECL defaults), 59% chose the dialog option that allowed the unsigned active content to run. So, in this situation, over half chose expediency over security. Again, this situation is quite close to “as good as possible” within the usability limitations of the deployed product. The user community was experienced in the technology, and had recently been warned of problems with open ECL defaults. In addition, virus warnings had for some time been prevalent in the media.

4.2. Other Issues Affecting Responses

Thirteen of the self-reported Gold Star users (15%) stated that they moved the ESA dialog aside and read the email message (Figure 5) before aborting the active content. While the text of a truly malicious virus would not make clear that the user should not execute the active content, often the text of recent virus emails triggers a cautious response in suspicious minds. Four of the Gold Stars indicated that they let one alert go and only caught it on the second. We would have preferred to have the active content issue only one alert, but it took at least two protected actions to send a simple email. This is perhaps more typically indicative of the number of alerts necessary for a malicious virus to do anything. It also allowed us to measure whether the unsigned active content was configured to execute, or whether the user was prompted before it completed.

One Gold Star reported that he had his mail database configured to not allow stored forms. Stored forms allow a document with a design different from the designs in the database to be placed in the database. The virus sent required the ability to attach active content to the PostOpen event of the form (design), and so required the use of stored forms. The feature to disallow stored forms provides extra security, since active content that is restricted to documents (not forms) can only be executed in Notes after the user takes some explicit action (such as pressing a button).

Several people complained that they still see many alerts. This is clearly unacceptable. Seeing many alerts in the course of a normal workday negates the whole purpose of having the ECLs in the first place. It encourages people to press “Trust Signer” or “Execute Once” only slightly less rapidly than the code would have executed unfettered. SoftwareHouse was not using a small identifiable set of signer names, and had no process to keep the Administrative ECLcurrent. One respondent said that they had hit trust signer literally hundreds of times after taking the Administrative ECL update when it was recommended slightly less than 3 months earlier.

Twelve people reported that they saw the sender's email message signature information, and so disregarded the unsigned nature of the alert. The Notes client displays a message in the message area at the bottom edge of the client when a signature is found on a document. A better

test would have been to send the unsigned active content in an unsigned mail message.

4.3. Miscellaneous Other Issues

At least 23 people pressed “Trust Signer” to the first alert. We know because we received more than one canned response message from them, since they read the mail message at least twice. The later ones had a much shorter time between actions than the first. This means that their ECLs became more open than they were before the survey. These users were encouraged to go to their ECLs and remove any permissions for the "-No Signature-" entry, and a general notice on the problem was posted in the SoftwareHouse discussion database. This was an unfortunate side effect of the survey.

Several people read the email on machines other than their primary work machine. They had tightened their ECLs on their workstation, but not, for example, on their laptop. Others mentioned that they now realized they needed to update their laptop. One person with open ECLs pulled the network plug when they realized what was happening after reading the email, then called to report their actions to the survey person. Some people were glad to get the survey and reported that they fixed their ECL settings. Others were not so happy (we received 4 flames).

A number of people were concerned that they had been previously unaware of this issue. Several of the more conscientious recipients were confused about whether they had done the right thing with the alerts and what their ECL should look like. Users who had done the right thing or who had secure ECL settings couldn’t tell and thought they needed to do something. Exercising the active content abilities of Notes to take a survey on their use had an additional the side effect. It made the issues around proper use of active content more obvious than previous the requests to press a button to ensure secure defaults.

One non-development person with much outside exposure asked for more documentation for lay users, and several other folks asked just what was considered to be best practices in this situation. Lotus Notes R5.02 featured a release note on some of these issues, and an article in Notes.Net, a technical resource for Domino administrators and designers, discusses how to deploy tighter ECLs [3]. Organizational best practice is to have a signing policy and a limited and identifiable set of identities for signing. IBM follows this practice. Groups developing third-party applications to run on Lotus Notes find it difficult to adhere to this discipline during internal development cycles.

Although the survey and subsequent discussions emphasized default settings for the LotusScript and @command portion of active content security, several recipients asked about the Java and Javascript ECL settings. The Javascript ECL allows “Default” and “No signature” entries full access to windows and URLs from the same host. This is because the Notes client can also be used as a web browser, but there is no standard for signing Javascript on the web. These defaults match default browser settings. In the Notes client, they can be reset by the user or by the site administrator via the Administrative ECL.

5. Recent Enhancements

Many of the issues with ECL use raised in this study and by Lotus Notes customers fall into two categories: getting the user's ECL to cover normal and acceptable use, and tracking and limiting the opportunities for users to make security decisions that they do not understand in the first place.

In the currently shipping version stream, the Refresh button in the ECL dialog box was added in R5.0.5 to provide a single consistent mechanism for users to refresh their ECLs (with the proper instruction). Previously, a button or similar mechanism would need to be coded into a mail message or other database document with the @RefreshECL command.

Domino 6 provides administrators with full control over updating client ECLs. Domino 6 administrative *policies* provide a mechanism to associate specific types of administrative information with individual users or groups of users. Security sub-policies let administrators define any number of named administrative ECLs, and associate them with different groups of users. An ECL policy consists of a reference to a fine-grained administrative ECL and an update policy. The update policy specifies both a frequency and a mode. The frequency of updating the user's ECL from their administrative ECL is either never, once daily, or when the administrative ECL has changed. The update mode offers the choice to refresh the user's ECL by merging in the administrative ECL (as described in Section 2), or to replace the user's ECL with the administrative ECL. The policy for updating a user's ECL is checked when that user authenticates to their home server, which contains their administrative ECL. These features ensure that user's ECL will stay in sync with the organizational policy in the administrative ECL. Administrators still have the challenge of ensuring that the administrative ECLs cover normal and acceptable cases. Giving in-house active content developers a separate namespace for them to use when signing active content that is distributed throughout the company is the most tractable policy. ECLs stay up to date with a single entry of the form "*/Signers/SoftwareHouse". Developers with the skills to sign code can easily change between their configured personal identity to the a trusted signing identity to re-sign the final version before sharing it by using the Notes "Switch ID" menu command and

authenticating against the second ID file. They can use Change Location to change their email address, if that is also needed.

Each administrative ECL contains a flag indicating whether or not client ECLs based on it can be changed by the user. Users who are not expected to run active content from sources not listed on their administrative ECL can have this flag set. If they receive malicious active content from an untrusted source, it will not be executed. If they receive useful active content from a source that they'd like to trust, they have to request that their administrator make the change to their administrative ECL.

The Notes 6 client logs ECL information, including any ECL alerts displayed to the user (and the user's choice of action) and any changes to the ECL. Figure 7 contains some example messages. Administrators can harvest this information to see what changes are necessary to make to the administrative ECLs, to gather information about users' behavior and to troubleshoot potential issues. Notes 6 also includes a new Multi-User feature that makes using Notes from multiple machines (such as home and work, or a laptop) a much more consistent experience. The users' personal changes to their configuration information, such as ECLs, is replicated for them from their client machine to their home servers. These changes are then replicated back down to the user's client when the user accesses the server from a different client installation.

01/09/2002 12:01:47 AM ECL Modification: CN=Lotus Notes Template Development/O=Lotus Notes was granted the right: Access to current database.(Using Workstation)
01/09/2002 12:01:47 AM ECL Alert Result: Code signed by CN=Lotus Notes Template Development/O=Lotus Notes was allowed to execute with the right: Access to current database.
01/09/2002 12:01:47 AM ECL Alert Details: DB Title: CDB, DB Path: c:\Lotus\Notes\Data\mail\cbassett.nsf, Design Note Type: Form, Design Note Title: Memo; Memo, Design Note ID: 2DE, ESA Description: NotesDatabase.GetProfileDocument Signature Status: No error.

Figure 7: Client ECL log

In addition to the features added to make ECLs more manageable and minimize the user interactions with them, several informational and design changes were made that make ECLs more usable. Figure 8 shows the Notes 6 layout of an execution security alert. Additional text explains the reason for the alert. The signer's identity is more prominent. The options are worded in a way that is easier to understand. It avoids the use of the term "Abort", which one test subject said sounded quite unpleasant; not like protection at all. The power user is given a "More Info" button. It provides extra information that is useful in the case of behavior that is suspected to be a bug.

Sometimes the features that are missing are as important as those that are present. Years ago, Lotus Notes had the ability to tunnel its rich mail format over the Internet when it knew the recipient was capable of processing it (by encapsulating and decapsulating at gateways). This feature was removed to prevent the more dangerous forms of active content attacks coming from outside. While it would be desirable to restore that functionality, this study provides evidence that it is not yet safe to do so.

6. Related Work

We are not aware of any other work that studies or evaluates the usability of active content security features.

Alma Whitten's bibliography on human factors in computer security [4] lists a number of references in the field. Just as our work evaluates how usable Notes ECLs are, [6] evaluates the usability of PGP encryption. It performs a case study of PGP 5.0 to determine whether it could be used by cryptography novices to achieve effective electronic mail security. A cognitive walkthrough analysis found a number of flaws, and laboratory user tests showed that the majority of subjects could not sign and encrypt a message in 90 minutes. The tests in [6] are more controlled and will therefore be easier to compare to other controlled tests. Our work is a field study, providing richness of feedback, but lacking the same level of control.

[8] applied user-centered design principles [7] to an authorization service for distributed systems. Contextual usability testing provided a rich picture of a security administrator's job. Formal usability testing provided direction on simplifying some features and making others more self-explanatory. Our work is closer to contextual usability than formal usability studies, since it occurred in the field, not in the lab. It does not provide the same concentrated detail that contextual interviews provide, but it does provide information over a much broader population than formal, structured lab usability testing.

[1] and [5] look at the socio-technical system of password choice and management. [1] provides suggestions for how to better inform users about password security, while [5] looks at the persuasive power of mechanisms, policies, tutorials, training and discourse. Some of our observed user reactions give indications about the socio-technical system of active content security, and may be useful in indicating future directions for study.

Figure 8: Notes 6 Execution Security Alert

The Notes ECL features we tested can be compared to the features in browsers to control the execution of active content. With version 5.5 of IE and continued in version 6, it is possible to designate active content sources, web sites, into several security zones. There are currently four of those; Internet, Local intranet, Trusted sites, and Restricted Sites. Security for each of these zones can be configured to one of the four templates, called High, Medium, Medium-low, or Low, or it can be customized in detail, starting with one of these templates. IE allows the user to configure whether to disable, enable, or prompt on the downloading of signed and unsigned ActiveX controls, as well as running ActiveX controls and plug in, and scripting ActiveX controls (marked safe or not marked safe). There are no finer grained permissions based on operation within these categories. It also allows the user to configure whether to enable, disable, or prompt when running unsigned or signed Java content, with a fine grained permission list based on Java 2. The Local Intranet zone defaults to Medium-low, which is the same as Medium without prompts. The prompts for code that cannot be signed lack the feature of allowing the user to change the behavior or zone for a site in the context of receiving the prompt (through that dialog). In addition, when a site has panes from multiple sources, it is difficult or impossible to determine the source of the script from the prompt. Users can choose to trust new publisher certificates in context, and revoke that trust later. They cannot chose to revoke the trust in certificates that are shipped trusted by IE. The Trusted zone is convenient for adding sites the user commonly visits and is familiar with. Some sites are not usable without allowing active content to run, so the user must trust them in order to use them. The Restricted zone takes some attention from the user to make use of. To minimize annoying prompts, regularly visited sites or ad sites that don't need active content to provide the features of interest to the user can be placed here. It is not clear how an administrator would configure the defaults for their site.

Netscape Communicator 4.73 allows the simpler options of enabling or disabling Java or Javascript.

7. Future Work and Conclusions

One area for future work is consideration of the appropriate granularity of permissions for what types of signed active content can be run. One "yes" should not give away everything forever. However, the finer

granularity is more confusing to users, and gives them less context and information to base a decision on. We believe that a two tiered level of granularity, similar to Domino ACL access levels and discrete permissions within those levels, would provide a starting point. Large grained levels such as "ability to read anything I can read" would make sense to many users. Allowing sites to customize the permissions within levels, or even define new levels, can provide administrators with desirable control, and users with still more meaningful granularity.

Since the study was a field study and not a controlled study, it raises a number of questions about what conditions affect the process of users deciding on their course of action when faced with a security-related choice. How would the results have differed if the users had not been warned of the problem several months in advance? How would the results have differed if the majority of the users were encountering the security choice for the first time? How would the results have differed after an actual outbreak of malicious active content from the same source application as the one we tested? A more scientific study could also vary how dire the warnings were. If the warnings got more and more dire, would users become concerned at a certain stage and choose not to execute potentially dangerous active content? Or would the warnings have to be extremely dire immediately to cause users concern? There seems to be very little work on allowing users to recover from a wrong decision. What effect would such features have, both on user actions and on the resulting security of the system? Studies involving multiple companies and different interfaces would also add a great deal of information to this area.

More research is needed in shielding users from having to make security-related decisions while still providing them with rich and flexible features. The Web-based client for Domino mail and calendaring, iNotes Web Access, filters and strips potentially harmful active content on the server side. Extensions to that might report the filtered active content to an operator, allowing the operator to vouch for it. User choices could be more constrained in more potentially dangerous situations. For example, Execution Security Alerts could be configured to not allow the user to execute the action if there is no signature information available, or if the signature is from a name not listed on the ECL or from a foreign domain (or from a domain not on a list of trusted domains).

The study occurred during the period of time when a product that was being shipped was changing its defaults on controlling active content from open to secured. When we enabled secure defaults in the development group at IBM, we initially found that some features placed in the product by groups other than the security group caused deployment problems for the secured active content default. Once all developers and testers were running with secured defaults, problems that arose between newly-developed features and secured configurations were found earlier and by the developers making the changes (who were in the best position to fix those problems). This argues for some stage of all development activities proceeding with the most tightly-secured configurations.

In this paper we detailed the features currently shipping and soon to be shipping in Lotus Notes that are used to protect users from potentially harmful active content in Notes databases (including their email database). We described the features available to administrators to configure user's active content security settings, the features available to users to view and modify them, and the interactions users are faced with when untrusted active content attempts to execute in their Notes client.

Our study shows that the majority of the users involved will take an explicit and straightforward one-time action to secure their electronic work environment when requested to, and directed by, a trustworthy colleague. When their work flow is interrupted with a security dialog, just under half of that same user population will make a choice that is unsafe, even after being instructed in the safe course of action 3 months earlier by the same trustworthy colleague.

The study makes it clear that the common software practice of warning users of danger but letting them click on something to proceed anyway is not going to provide adequate security unless the either the user community or security-related interfaces undergo some radical sea change. This change may include education, or better and more pertinent information from the software. The more frequently security warnings appear in everyday use, the more users will learn to click "OK" without thinking or even remembering that they have done so. False alarms should be treated as serious security vulnerabilities instead of acceptable irritants. Until we are able to write software applications that can better tell the difference between safe and unsafe, taking the "safe" option should be made routine by both the culture and the software, and taking the "unsafe" option should be audited and reviewed.

8. Acknowledgements

This study would not have been possible without the time, skills, and experience of Chuck Bassett, and without the support of management in SoftwareHouse. We also thank Amy Smith for her editorial assistance.

9. References

[1] Anne Adams and Martina Angela Sasse. Users Are Not The Enemy: Why users compromise security mechanisms and how to take remedial measures. Communications of the ACM, 42 (12), pp. 40-46 December 1999.

[2] CNET, Gates memo: "We can and must do better". http://news.com.com/2009-1001-817210.html?legacy=cnet

[3] Amy E. Smith (with Charlie Kaufman, Chuck Bassett and Mary Ellen Zurko). Staying Alert with Execution Control Lists. http://notes.net/today.nsf/f01245ebfc115aaf8525661a006b86b9/3a9da544637a69b2852568310078b649?OpenDocument.

[4] Whitten, Alma. References for Human Factors in Computer Security. http://www.sims.berkeley.edu/~alma/biblio.html.

[5] Dirk Weirich and Martina Angela Sasse. Pretty Good Persuasion: A First Step towards Effective Password Security in the Real World. Proceedings of New Security Paradigms Workshop, September 2001.

[6] Alma Whitten and J.D. Tygar, Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. Proceedings of the 8th USENIX Security Symposium, August 1999.

[7] Mary Ellen Zurko and Richard T. Simon, User-Centered Security. New Security Paradigms Workshop, 1996.

[8] Mary Ellen Zurko, Richard T. Simon and Tom Sanfilippo, A User-Centered, Modular Authorization Service Built on an RBAC Foundation. Proceedings of IEEE Security and Privacy, 1999.