Project 1: Documentation and Grading

CS 161

October 13, 2006

Please read this early so that you have a good understanding of what we expect.

1 Documents

As specified in the project handout, there are a few different documents you should turn in as part of your submission. Here we'll try to be a bit more explicit about what we are expecting. You are to hand in (via the electronic submission process) the following documents:

Final Design Document If you have not changed your design since the design review, this could theoretically be the exact same document you submitted for that. If you have made minor changes, please highlight them or provide some other easy way for us to find all the changes. If you have made major changes, please indicate which sections have changed and which have not, and include some text (perhaps in its own section, perhaps a bit in each changed section) giving a (very short) high-level overview of the changes.

Security Analysis Document This is the most important part of your documentation. In here is where you will convince us that your system achieves the security goals, with specific reference to your implementation. More about this below.

README file This is just a place to put practical information on how to run your code. Please tell us exactly how to run your server, how to set up your clients so they know the server's IP address, what arguments we need to pass (if any), and so on.

The final design document should be about the same length as the original (4000 words max). Please also use 4000 words as a limit for the security analysis document. This limit is somewhat flexible; if you find it too restrictive, let your TA know. We suspect that the security analysis documents will be shorter, however; a 1500-word document that is well argued and thorough would be great. (Hint: As long as nothing important is missing and everything is clear, graders strongly prefer to read short solutions.) There is no set limit for the README, but by nature it should be very short.

2 Security Analysis Document

The goal of this document is to convince us that your system meets the security requirements. You should include enough low-level details to demonstrate that you have used cryptographic tools correctly.

For example, let's say that your assignment was simply to write a program that encrypts the string "Hello world" and sends it over the network to a friend sharing a symmetric key with you that no one else knows; the only security requirement is confidentiality. Here is how you might address the requirement in your document:

> To protect the confidentiality of the message, we use AES in cipher block chaining (CBC) mode. In our code, the only place where anything is sent over the network is in the `sendMessage()` function, and in that function we encrypt the message right before sending it over the socket. So there is no way for the message to go over the network without being encrypted. Now the problem states that our key is shared with the friend and no one else knows it, so only the friend can decrypt the message. There are no known attacks against AES better than brute-force, and a brute-force attack is infeasible with a 128-bit key, so the confidentiality of the message is assured.

Your problem is more elaborate and your systems are much more complicated, so you will have to take more considerations into account, but this is the basic idea of the kind of analysis we're looking for. Here is one more example. Let's say that the problem is to get the time of day from a time server and be sure that it's fresh; you may assume the server is trusted and you have its public RSA key. Here is how you might address this problem:

> Our SuperTimeLookup(tm) protocol specifies that the client generates a random 128-bit number as a nonce and sends it to the server (the *request message*). The server then replies with the nonce plus time of day, and a signature over the two values. If the signature checks out, then the client knows that the server must have seen the nonce and generated the message, because only the server has the private key that signed the message (including the nonce). Furthermore, the time value must be fresh because the nonce was generated randomly and the space of values is too big for an attacker to guess the right value, so no replay attack is possible (if we used an 8-bit number, for example, the attacker could have generated all 256 possible nonces and sent them to the server yesterday, then replied to our request message with the appropriate one today, giving us the wrong time). Therefore we are guaranteed that, if the signature is valid, the time given in the message is the current time (i.e. no earlier than when the request message was sent).

These examples should give you a good idea of what we're looking for; the first one specifies how the code uses a cryptographic primitive to achieve confi-

dentiality, and the second one describes the use of a nonce in a message protocol to guarantee freshness (which is tied to message integrity and/or authentication, depending on exactly what you're trying to accomplish). Again, of course, your analysis will be longer and more in-depth because your protocols and systems are more complex, but the idea is the same.

2.1 Denial of Service

You are not required to consider Denial of Service attacks when implementing your system, but here we do want you to think about them and discuss how your system would fare under such an attack and how it could be improved. The project handout explains what we expect for this. It can be one section of your security analysis document.

3 Grading

3.1 Testing

We will test all the required functionality of your system. Things that are not required, such as notification of sign-on/sign-off, do not need to be implemented, but you need to implement enough to make sure your system is usable at a basic level. (If there is anything that might trip us up when we try to run your system, please mention it in the README.) We won't be taking off points for occasional code crashes when testing corner cases, but your code shouldn't be crashing all the time; it should fail gracefully from error conditions.

We may do some small amount of (currently unspecified) security testing of your code. You are not required to do security testing yourself, so you will not lose points because we discover something that could only be discovered by testing. One reason we might test is if we notice what seems to be a weakness in your system; we may then test it to see if an attack does work in practice. Or if some part of your code is obscure to us and we can't quite tell what it does, we may find it easier to test it rather than puzzle over the source. But please do make your code as readable as possible!

3.2 Analysis

The primary means we will use to grade the security of your project is reading and evaluating your security analysis document. We will look to make sure you address all the security requirements wherever they are relevant. We will look to see if you defend your choice of cryptographic tools and your use of them. We will especially look at your analysis of message protocols and why you believe them to be secure. For full points, you need to be thorough in your assessment of the system and precise in your arguments. There is necessarily a subjective component to this grading, but you can turn that to your advantage by giving clear, forceful, well-reasoned arguments.

You won't lose points if there is some obscure attack that you didn't think of, or if one of your arguments turns out to be wrong because of something outside the scope of this class. You will be marked down if you make faulty arguments, or if you miss obvious attacks that have been mentioned in this class, for example.

We will look at your source code, but only when necessary; the security analysis document should contain your full analysis with reference to source code when appropriate. If the analysis doesn't seem to match the code, or if the code doesn't seem to implement what the design document says it should, then of course we will take points off for that as well.

We will be very concerned about being reasonable and not holding things against you that you have no way of knowing, but we do expect a high level of attention to clear writing and well-reasoned, supported arguments.