This is an example of a set of test strategy ideas. It's a raw set of rough notes from a one-hour brainstorm I did in 2018 on how I might think about testing a login process. It's intended as somewhat disgruntled reply to those examples of a script "testing" a login page by checking for the 'you are now logged in' message. There are lots of things that can go wrong with the login process, and around it. A couple of things to mention. First, these are rough notes; very rough. I have barely cleaned them up; just enough to Forgot username make them semi-readable. To be useful, they would need refinement, or to be tossed out and replaced with something clearer and cleaner. On the other hand, it's a start. -Authorised user will not be able to log in The second thing to mention about these notes is that they're incomplete! They are *by no means* done. You'll notice tons of empty nodes and dangling ideas on the map. There could be lots more things to question or to cover in some context or another. Hacker will be able to log in Brute force approach Third, I'm not a expert in aspects of testing (particularly security and performance) that would require deep expertise here. I'm not doing this stuff every day. If I were, you can bet that I would immerse myself in those aspects of testing and risk investigation. I'd hang out with performance testers and security testers and programmers, and I'd collect stories about risks and problems from the Net. Unauthorised user will be able to log in Previously authorised user will be able to log in Hacker will be able to get access to the system while evading the login process, or some aspect of it System is vulnerable to some form of attack (other than unauthorized login) by exploiting vulnerabilities in the login interface Fourth, to answer the inevitable question, No, we wouldn't consider all of these things on every product, and certainly not on every build. But we might choose to pull out some of these ideas and use them to guide or influence our testing from time to time. Username/password will be intercepted SQL injection The key in all this is to recognize two contexts for testing. In some cases, we do want to test quickly, in a shallow but useful way, when that helps us to maintain a good pace of development and not slow down. -Error messages will reveal useful information to hacker In other circumstances, we may need to test deeply to look for subtle, hidden, rare, intermittent, emergent bugs User will be fooled into revealing credentials

Username/password/personal information will be leaked via vulnerability

Logout or unsuccessful login will leave system in an unstable or insecure state Deep testing, however, is time-consuming and disruptive to programming work, so we don't even try to do it all the time, or on every built of the properties simultaneous logins It may be possible to swamp the system with going beyond limits of account management r Single sign-on issues

User may not be able to access other systems or organ What is the set of acceptable of Back doors of which we may be unaware Record of user access will not be retained What characters are acceptable, but must be escaped? If we're testing for these, test for one at a Failed attempts to login will not be retained ____Especially important WRT to tracking hack attempts Site looks sufficiently amateurish that user won't risk a login attempt (original Verified by Visa was like this) is there filtering for that? To what degree are we monitoring the server? What happens on intermediate machines? Could login data be intercepted?

Are username and password encrypted? Which characters are User ID only? Minimum and maximum lengths Both accepted? Audio Captcha Test ideas Two-factor authentication? Huge input data Zero input data Single sign-on? OAUTH? W Will this be authenticating with another system (like those Twitter add-ins)? Combinations of valid and invalid usernames/passwords/CAPTCHAs/security questions Maximum length Am I the only tester on the project? The only tester focused on this feature? Am I qualified What happens before, during, and after the process on the client? Does the system properly block lapsed accounts? What happens before, during and after the process on the server? What's the protocol? Is it subject to some kind of standard?
Where can I learn about the protocol? Login attempt when locked out What is the lockout strategy, if any? What encryption is in use? You're not really using GET, are you? Quality Criteria -® Login after a timed-out session Login after a rejected login What happens or Is there a flow model or diagram anywhere? logging in? intermediate systems? User ID Does the user have access to resources across two sessions What do I know about? Are cookies being used? Cookies What do they contain? Are their contents private? Should they be? Assign access token to process or thread Compare credentials to authentication database Connection to "external" systems See "Product Elements" of the HTSM About indvidual login activity? What information is gathered and tracked? Are there other systems accessible via single sign-on (SSO?)

Is the user able to access everything that he should?

Is the user prevented from accessing the things that See standards related technologies? documents, RFCs, etc. Bearer Token Usage User ID XSS, CSRF? Restricting group identifiers HTTP header injection? Email injection? Default owner, primary group, access control list for objects created by the subject asso When a message box is displayed? How do we determine when the user is logged in and the system is ready for use? Has the login been recorded? What information is kept in the record? Other tests or checks? e.g. OAUTH Through a third party? Operating systems User appropriately authorized? Is anything being updated or pushed at Login scripts finished? Okay, not this time. Ever? Login recorded in the access logs? Is that hannening? Are they well-structured, Is the log secure? Does the log contain sensitive information What happens under lots of load? What triggers them? What might not get cleaned up if a user can get here but can't get past here Has anyone else tested this? Can that testing be trusted? Are there limits on the number of logged-in users? Are there limits on invalid login attempts? -How is the state of a logged-in user identified and maintained: