

Name: Acunetix AspNet									
URL: http://testaspnet.acunetix.com/									
Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes
Cenzic Vulnerability Claims									
FP	SQL	/Comments.aspx	id (POST)	N	N	Y	Y		Rejected/FP: These are not attackable over POST. Going to the page without a proper id value on the GET will always give an error response, but you cannot put an attack payload in the POST id value. Try /Comments.aspx?id=1%20and%201=1 this will work, but if you try as a POST it will not. We report these as SQL Error, meaning that we see evidence of causing an error in SQL processing and causing a disclosure to that effect. They are certainly not false positives because it is clear that there is an inappropriate disclosure of a stack trace and db errors. While not a false positive, perhaps there is debate about whether it counts as an exploitable SQL problem, but I believe that such errors are SQL Injection related disclosures that are important for users to fix. I've attached the HTTP response at item #1 in responses list. OK. Fair enough, I now agree that this can be removed as an FP. But it is not an error caused by any injection, it is basically just an error that is already on the site.
FP	SQL	/ReadNews.aspx	id (POST)	N	N	Y	Y	Claim: New vulns everyone else missed	
Rejected	App Exception	/Comments.aspx	_EVENTVALIDATION	N	N	Y	Y	Rejected: These are just viewstate errors. There is no vuln other than standard error output. I do not consider this worth including, because its a waste of time. Standard error output can still be a vulnerability. Granted these are viewstate errors, but they are showing a stack trace and associated inappropriate disclosures. It even says in the response (item #2) that this error page might contain sensitive information because ASP.NET is configured to show verbose output. This is an important part what App Exception vulnerabilities are about. I guess we can just disagree to some extent. I do think people must turn off this verbose reporting, but with ViewState its possible to see this on every single URL when verbose reporting is on. So it should be reported at least once, but I still dont think it is all that interesting and worth including in my list.	Claim: New vuln everyone else missed
Rejected	App Exception	/Signup.aspx	_EVENTVALIDATION	N	N	Y	Y		
Rejected	App Exception	/Login.aspx	_EVENTVALIDATION	N	N	Y	Y		
Rejected	Frame Injection	/ReadNews.aspx	NewsAd	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: This is just another variation of XSS attack listed already. Once you have XSS, then most of the time you can add an iframe. Why should this be considered as an additional attack? We did not leverage XSS, and we are not adding an iframe, this attack simply supplied an alternative URL to a GET parameter that normally has a URL as its value, and the app is not validating the URL. This is a form of remote file inclusion, and you accepted this class of vulnerability from NTO, even for this target. I've attached the request and response as item #3. You can report it as remote file inclusion if you wish. I still say this is nothing more than a duplicate. You did identify a good way to use it to make the injection point (which is the iframe src) point to a bad site. However its still the same issue. It certainly is not remote file include, because that category of attack is intended for situations where you can make the server include a remote file, such as the case with PHP include file attacks.
Non-Disputed Vulnerability Counts from Original Scan									
Verified	Count of all other original issues			5	5	N/A	N/A		
FP	Count of all other original issues			0	0	N/A	N/A		
	Valid Vulnerabilities			Possible	7	7	7	7	
				Found	5	5	5	5	
				Missed	2	2	2	2	
	False Positives			Reported	0	0	0	0	
Name: Acunetix TestPHP									
URL: http://testphp.acunetix.com/									
Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes
Cenzic Vulnerability Claims									
Accepted	XSS	/userinfo.php	uaddress	N	N	Y	Y	Claim: Now finds	OK
Accepted	XSS	/userinfo.php	urname	N	N	Y	Y	Claim: Now finds	OK
Accepted	XSS (persistent)	/userinfo.php	urname	N	N	Y	Y	Claim: Now finds	OK - NTOSpider also flagged this as persistent
Accepted	XSS	/userinfo.php	uphone	N	N	Y	Y	Claim: Now finds	OK
				Y	Y	Y	Y	Dispute: We don't see this reported.	Rejected: I did see vulns like /404.php?blah=<script>alert(345)</script> The 404 page does display the URL you entered but the string is being escaped and I could find no way to break out, and the attack strings from the results did not execute either. So this was a FP, and there were many instances of this but I only counted it as one. OK
FP	XSS	/404.php	The URL Path						
Non-Disputed Vulnerability Counts from Original Scan									
Verified	Count of all other original issues			10	10	N/A	N/A		
FP	Count of all other original issues			2	2	N/A	N/A		
	Valid Vulnerabilities			Possible	28	28	30	30	
				Found	10	10	10	10	
				Missed	18	18	20	20	
	False Positives			Reported	2	2	2	2	
Name: Cenzic Crackme									
URL: http://crackme.cenzic.com/									
Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes
Cenzic Vulnerability Claims									
Pending	SQL	/Kelev/php/approvalloanpagedetail.php	hRequestId	N	N	Y	Y	Claim: New vuln everyone else missed	Unfair?: The other scanners were run using maryblah/mary123 which does not seem to have this functionality. So I am not sure I can fairly include this without re-running all the other scanners with admin/adminm3 Your original report shows using admin/adminm3 in addition to maryblah - so I'm not sure I understand what was done originally and what the issue is. My mistake in the spreadsheet. I knew that the admin/adminm3 account existed and had that note in my file, but I didnt use it for any of the scans, I only used maryblah/mary123 Considering that it would be unfair to include this when it only works under the admin/adminm3 account.
Accepted	XSS	/Kelev/php/approvalloanpagedetail.php	hRequestId	Y	N	Y	Y	Claim: Now finds	OK
Accepted	XSS	/Kelev/view/updateloanrequest.php	drpLoanType	N	Y	Y	Y	Claim: Now finds	OK
Accepted	XSS	/Kelev/php/loanrequestdetail.php	hUserId	N	Y	Y	Y	Claim: Now finds	OK
FP	SQL	/Kelev/view/rate.php	User-Agent (Header)	N	Y	Y	Y	Dispute: We don't see this reported.	I did see it but dont have the details anymore, so will remove it.
FP	SQL	/Kelev/php/loanrequestdetail.php	hRequestId	Y	N	Y	Y	Dispute: We see messages regarding MySQL and mysql_fetch_array()	Rejected: The reason this is an FP is that the hRequestId has nothing to do with generating the SQL error. The error strings about mysql fetch array are already on the page regardless of any attacks performed. Try with (maryblah/mary123) I also see this error in responses for other attacks like XSS, but in these cases we also happen to break the query. To further verify we have carefully examined how the app is handling the input and what it does for example when the query using the inputs does not find results. They only case we see that causes this error output is when we break the SQL query syntax - which happens due to various injections. We do succeed in infiltrating the SQL query and affecting the syntax. Again, I can agree that this can be removed as an FP since you are identifying that a SQL error is on the page. But it is not caused by the attack strings mentioned... they are just already there.
FP	XSS	/Kelev/view/updateloanrequest.php	txtEmail	N	N	Y		Dispute: We prove javascript execution through an alert popup	Pending: In my review it appeared that these two fields are not vulnerable. What happens is that they show up when you attack all the inputs at once and do not properly identify which input was the one that was vulnerable. Please provide an attack string where only txtTelephoneNo has an attack string and is vulnerable with an alert that pops up so I can see this for myself. The cases I see reported are single field injections. I've put the requests and responses in items #4 and #5. Interesting. From your traffic it appears that what you did worked. I will manually confirm, but it looks like you are right.
FP	XSS	/Kelev/view/updateloanrequest.php	txtTelephoneNo	N	N	Y			
Rejected	Brute Force		(no logout)	N	N	Y	Y		I did not include this in my test. It is not a vulnerability, but rather a best practice issue. There were too many Best Practice issues to reasonably include. I'm not sure how you distinguish between vulnerabilities and best practice issues. In your paper you list "Brute Forcing" as an issue, and lack of logout is the key vulnerability that allows Brute Forcing. With no logout, and enough time, any username and password can be figured out (even faster when there are no password strength requirements). It is exploitable, we just did not run long enough to find acceptable real usernames and passwords. I agree that logout limits are important and it really is a best practice that should be followed, but it is not in an of itself a vulnerability that can be exploited right away. Which is why I dont agree to including it in the results.
Non-Disputed Vulnerability Counts from Original Scan									
Verified	Count of all other original issues			10	12	N/A	N/A		
FP	Count of all other original issues			2	1	N/A	N/A		
	Valid Vulnerabilities			Possible	17	17	18	18	
				Found	10	12	10	12	

				Missed	7	5	8	6			
		False Positives		Reported	2	1	2	1			
Name:	HP zerowebappsecurity										
URL:	http://zero.webappsecurity.com										
Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes		
Cenzic Vulnerability Claims											
Accepted	XSS	/cookieest/ShowCookies.asp	FirstCookie (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	Keyed (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	Second (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	status (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	userid (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	username (Cookie)	N	Y	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/cookieest/ShowCookies.asp	(added cookie)	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: Just another duplicate Well, seems a enough different from just another dup that it is worth noting. Yeah, worth noting as another duplicate		
Duplicate	XSS	/cookieest/ShowCookies.asp	ASPSESSIONID	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: Just another duplicate Agreed. In my differences spreadsheet I marked this "Ydup" which means I acknowledged as a duplicate. Right		
Pending	XSS	/join.asp	address2	N	N	N	Y	Claim: New vulns everyone else missed	Pending: I do not see how you can find this page from normal crawling. In NTO's response they detailed the steps to finding the join1.asp, I would need to have the same kind of information from you as to how you would have found this page. I also did not train the other scanners to this page, so it seems unfair to have yours trained to it and not the others. So I dont think I can include this without re-running the other scanners with them trained to this page. As NTO noted, they got to the join.asp page from the login.asp page, and then they found vulnerabilities by submitting the form on the join.asp page to the join1.asp page. The reason Hailstorm found vulnerabilities on the join.asp page, is that an alternate URL for join.asp was observed with added injectable parameters. This was observed that when the form on the join.asp page is only partially filled. In this situation the submission to join1.asp page redirects back to the join.asp page, but now with injectable query string parameters. What was originally just "...join.asp" is now "...join.asp?name=&email=&surname=&house=" etc., and is vulnerable. I will investigate further, but your description seems plausible. To be fair, since your tool found this due to training, I should train the other scanners to this page as well to see if they would find these vulns, once they know about the page.		
Pending	XSS	/join.asp	country	N	N	N	Y				
Pending	XSS	/join.asp	email	N	N	N	Y				
Pending	XSS	/join.asp	homephone	N	N	N	Y				
Pending	XSS	/join.asp	house	N	N	N	Y				
Pending	XSS	/join.asp	mobilephone	N	N	N	Y				
Pending	XSS	/join.asp	msg	N	N	N	Y				
Pending	XSS	/join.asp	name	N	N	N	Y				
Pending	XSS	/join.asp	postcode	N	N	N	Y				
Pending	XSS	/join.asp	street	N	N	N	Y				
Pending	XSS	/join.asp	surname	N	N	N	Y				
Pending	XSS	/join.asp	town	N	N	N	Y				
Accepted	XSS	/pcomboindex.asp	referer (Header)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	sessionid (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	State (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	user-agent (Header)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	userid (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	username (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Rejected	XSS	/pcomboindex.asp	status (Cookie)	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: Just another duplicate		
Duplicate	XSS	/pcomboindex.asp	sessionid (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Duplicate	XSS	/pcomboindex.asp	State (Cookie)	N	N	Y	Y	Claim: Now finds	OK		
Accepted	XSS	/pformresults.asp	dbConnectString	N	N	N	Y	Claim: Now finds	OK		
Accepted	XSS	/pformresults.asp	txtFirstName	N	N	N	Y	Claim: Now finds	OK		
Accepted	XSS	/pformresults.asp	txtLastName	N	N	N	Y	Claim: Now finds	OK		
Rejected	Brute Force		(no lockout)	N	N	Y	Y	Claim: New vuln everyone else missed	I did not include this in my test. It is not a vulnerability, but rather a best practice issue. There were too many Best Practice issues to reasonably include. I'm not sure how you distinguish between vulnerabilities and best practice issues. In your paper you list "Brute Forcing" as an issue, and lack of lockout is the key vulnerability that allows Brute Forcing. With no lockout, and enough time, any username and password can be figured out (even faster when there are no password strength requirements). It is exploitable, we just did not run long enough to find acceptable real usernames and passwords. I agree that lockout limits are important and it really is a best practice that should be followed, but it is not in an of itself a vulnerability that can exploited right away. Which is why I dont agree to including it in the results.		
Non-Disputed Vulnerability Counts from Original Scan											
Verified	Count of all other original issues			9	24	N/A	N/A				
FP	Count of all other original issues			0	0	N/A	N/A				
				Possible	31	31	31	31			
				Found	9	24	9	24			
				Missed	22	7	22	7			
				Reported	0	0	0	0			
IBM Testfire											
Name:	http://demo.testfire.net										
URL:	http://demo.testfire.net										
Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes		
Cenzic Vulnerability Claims											
Accepted	Error/XPath Injection	/bank/querypath.aspx	SOAP: TextBox1	N	N	Y	Y	Claim: Now finds	OK		
Accepted	SQL Auth Bypass	/bank/login.aspx	uid	N	N	Y	Y	Claim: Now finds	This had been found and listed SQL Auth Bypass, which Hailstorm originally missed. The only account I know of that works on this site is jsmith/Demo1234 which is specific to this site and if that is all you found it must be in your database for this specific site, so is not realistic to include. We did not find via Brute Force. The vulnerability is SQL Auth Bypass which is based on SQL Injection technique and is credited to NTO, Appscan and Qualys. In the spreadsheet you gave me, it said "Brute Force Login". If you meant SQL Auth Bypass, then that is fine and should count		
Accepted	SQL	/bank/login.aspx	passw	N	Y	Y	Y	Claim: Now finds	OK		
Accepted	SQL	/bank/login.aspx	uid	N	Y	Y	Y	Claim: Now finds	OK		
Accepted	SQL	/bank/login.aspx	btnSubmit	N	N	Y	Y	Claim: Now finds	OK		
Rejected	XSS	/bank/customize.aspx	lang (POST)	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/FP: This is a new FP. This lang parameter is not attackable via POST. Please provide HTTP traffic that supports your case, because my manual testing does not show this to reflect the value at all. I've included the request and response as item #6. When rendering the response I can see the alert re-execute. It seems pretty clear from what I see. In your traffic you do have the method set as POST, but the attack is on the GET string and not in the post data section. So to be fair I will agree that this is not a new FP, but it is also not another new vuln.		
Accepted	SQL	/bank/ws.aspx	SOAP: creditAccount	N	N	Y	Y	Claim: Now finds	OK		
Accepted	SQL	/bank/ws.aspx	SOAP: debitAccount	N	N	Y	Y	Claim: Now finds	OK		
Accepted	XSS	/subscribe.aspx	txtEmail	N	Y	Y	Y	Claim: Now finds	OK		
Rejected	App. Exception	/bank/querypath.aspx	_EVENTVALIDATION	N	N	Y	Y	Claim: New vuln everyone else missed	These are just viewstate errors. There is no vuln other than standard error output. I do not consider this worth including, because its a waste of time. Standard error output can still be a vulnerability. Granted these are viewstate errors, but they are showing a stack trace and associated inappropriate disclosures. It even says in the response (item #7) that this error page might contain sensitive information because ASP.NET is configured to show verbose output. I would propose that this is what App Exception is about. I guess we can just disagree to some extent. I do think people must turn off this verbose reporting, but with ViewState its possible to see this on every single URL when verbose reporting is on. So it should be reported at least once, but I still dont think it is all that interesting and worth including in my list.		
Rejected	Brute Force Login		(no lockout)	N	N	Y	Y		Claim: New vuln everyone else missed	I did not include this in my test. It is not a vulnerability, but rather a best practice issue. There were too many Best Practice issues to reasonably include. I'm not sure how you distinguish between vulnerabilities and best practice issues. In your paper you list "Brute Forcing" as an issue, and lack of lockout is the key vulnerability that allows Brute Forcing. With no lockout, and enough time, any username and password can be figured out (even faster when there are no password strength requirements). It is exploitable, we just did not run long enough to find acceptable real usernames and passwords. I agree that lockout limits are important and it really is a best practice that should be followed, but it is not in an of itself a vulnerability that can exploited right away. Which is why I dont agree to including it in the results.	
Non-Disputed Vulnerability Counts from Original Scan											
Verified	Count of all other original issues			8	10	N/A	N/A				
FP	Count of all other original issues			0	0	N/A	N/A				
				Possible	24	24	24	24			
				Found	8	10	8	10			
				Missed	16	14	16	14			
				Reported	0	0	0	0			

Status	Vuln Type	File	Parameter	PaS	Trained	PaS	Trained	Vendor Notes	Larry Suto Notes
Name: NTO Webscantest									
URL: http://www.webscantest.com/									
Cenzic Vulnerability Claims									
Accepted	Auth Testing	/login.php (admin/admin)	login	N	N	Y	Y		See note below for Brute Force Login
Accepted	Command Injection	/osrun/whois.php	domain	N	N	Y	Y	Claim: Now finds	OK
Accepted	XSS	/404 handler (any random page)	referer (Header)	N	N	Y	Y		See next note
Rejected	XSS	/rfplaces/function.require-once	referer (Header)	N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: These are all duplicates of "/404 handler (any random page)" because those pages dont exist and show the custom 404 response which is already defined as vulnerable. I will add credit to the PaS result for that based them now finding these (OK, I did not notice they are really 404)
Rejected	XSS	/rfplaces/function.require-once	referer (Header)	N	N	Y	Y	Claim: New vuln everyone else missed	
Rejected	XSS	/gonowhere	referer (Header)	N	N	Y	Y	Claim: New vuln everyone else missed	
Rejected	XSS	/picshare/uploadsafe.php	referer (Header)	N	N	Y	Y	Claim: New vuln everyone else missed	
Accepted	XSS	/picshare/upload.pl	SESSIONID_VULN_SI	N	Y	Y	Y	Claim: Now finds	OK
Accepted	XSS	/bjax/servertime.php	msg	N	N	Y	Y	Claim: Now finds	OK
FP	SQL	/payment_analysis/checkdata.php	alpha_only	Y	N	Y	N		Rejected: These are still considered FP's because your scanner is telling the user that they can perform SQL injection against these when that is not correct. These are only app errors. We report these as SQL Error not SQL Disclosure, meaning that we see db related error information (codes 22007, 22008, etc). This class of finding is not a claim of proven exploitability, so, as such, the only question here is whether there is sufficient reasonable information to claim an inappropriate disclosure of db errors that might be used for further attempts to attack. Given that the responses show db level error codes, and this in a SQL Error SmartAttack, I think we found what qualifies, and therefore these are not FP. I agree that we cannot count twice though, so removing as FP but not adding as a found vulnerability makes sense. I understand now. The errors on that page are apparently hardcoded and look too much like stuff that would come from SQL server. So I agree these should not be considered FP's but also not vulns.
FP	SQL	/payment_analysis/checkdata.php	letters_only	Y	N	Y	N		
FP	SQL	/payment_analysis/checkdata.php	number	Y	N	Y	N		
								Disputed: These are vulns, and already counted as App Exception	
FP	SQL	/osrun/whois_nv.php	domain	Y	Y	N	N	Dispute: We don't see this reported.	OK - I did see these, but I will assume your update correctly avoids this now
FP	SQL	/osrun/whois.php	domain	Y	Y	N	N	Dispute: We don't see this reported.	
Rejected	Brute Force Login	/login.php (admin/admin)		N	N	Y	Y	Claim: New vuln everyone else missed	Rejected/Duplicate: This is already listed as Auth Testing (admin/admin) which you had missed. I will add credit to that above. (Yeah, I was not sure how to credit.)
Rejected	Brute Force Login	/login.php	(no lockout)	N	N	Y	Y		I did not include this in my test. It is not a vulnerability, but rather a best practice issue. There were too many Best Practice issues to reasonably include. I'm not sure how you distinguish between vulnerabilities and best practice issues. In your paper you list "Brute Forcing" as an issue, and lack of lockout is the key vulnerability that allows Brute Forcing. With no lockout, and enough time, any username and password can be figured out (even faster when there are no password strength requirements). It is exploitable, we just did not run long enough to find acceptable real usernames and passwords. I agree that lockout limits are important and it really is a best practice that should be followed, but it is not in an of itself a vulnerability that can exploited right away. Which is why I dont agree to including it in the results.
								Claim: New vuln everyone else missed	
Non-Disputed Vulnerability Counts from Original Scan									
Verified	Count of all other original issues			18	29	N/A	N/A		
FP	Count of all other original issues			3	2	N/A	N/A		
			Possible	47	47	47	47		
			Valid Vulnerabilities	Found	18	29	18	29	
				Missed	29	18	29	18	
			False Positives	Reported	3	2	3	2	