Hewlett Packard Enterprise

Performance metrics for .NET

HPE Security Fortify Application Defender

Contents

Technical white paper

Abstract

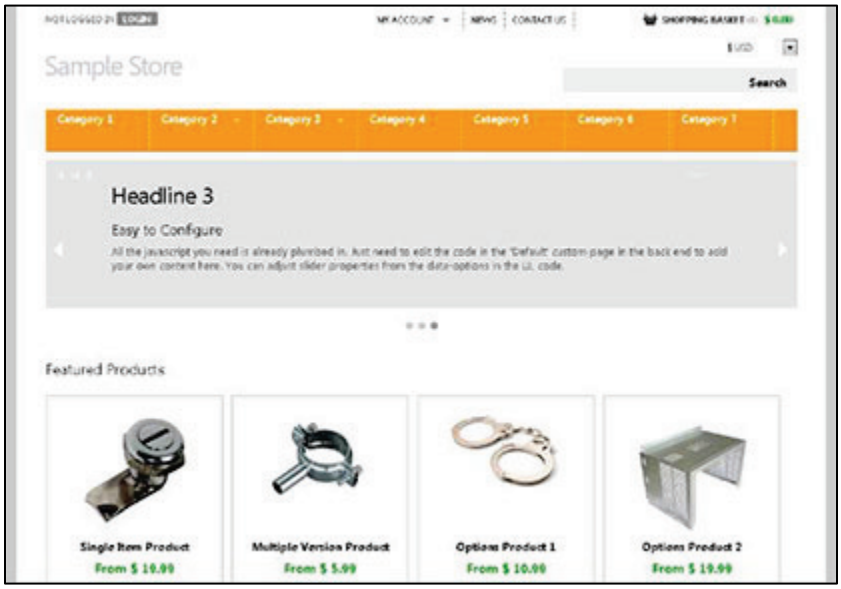
For every IT person, understanding the performance impact of adding a new solution to the production environment is invaluable to keeping servers up and running. To estimate the performance overhead incurred by introducing HPE Security Fortify Application Defender (.NET), we tested the application self-protection solution on a well-used, open source e-commerce application. Statistics show the execution time overhead range from 7 percent to 20 percent while the memory overhead is about 140 MB.

Evaluation

The evaluation of the Application Defender agent (.NET) was conducted on Kartris, a popular e-commerce application. Kartris was run in an environment that was dedicated to performance testing. HPE LoadRunner was used to gather performance metrics. In this section, we describe the different aspects of the evaluation.

Real-world, open source application

Kartris is one of the most popular open source e-commerce systems written in ASP.NET 4.5 with an MS SQL database back end. Kartris has a modern, user-friendly interface that makes use of AJAX to deliver a fast, feature-rich experience that works on all modern browsers, and includes mobile and tablet support. Out of the box, it is capable of handling sites with more than 1 million products. Kartris is available under an open source license, GPL version 2.



Evaluation goals

The goal of the test was to determine the performance overhead that Application Defender (.NET), an application self-protection solution, causes on the test application. We measured performance impact on server execution time and memory consumption.

Application Defender supports two different rulepacks: Application Protection Rulepack and Application Logging Rulepack. The Application Protection rules are designed to detect and protect from malicious attacks like SQL Injection, Cross-site Scripting, Command Injection, etc. Meanwhile, the Application Logging Rulepack is designed for auditing security related events like user logon, user logoff, read/write a file, etc. In order to measure the performance impact under different configurations, we measured six different scenarios:

Under normal traffic

1. Protection rulepack
2. Logging rulepack
3. Protection and logging rulepack

Under 5% attack traffic

4. Protection rulepack
5. Logging rulepack
6. Protection and logging rulepack

Lab environment

The test machine was running HPE LoadRunner version 12. This was the only machine that was connected to the external network. The HPE LoadRunner machine contained another physical network card to access the internal network set up for the test. The internal network contained two additional physical machines, one to run the Kartris website and the other to run the HPE Application Defender event collector channel. The Kartris application ran on Microsoft® Windows® 7 with SQL Server 2008 Express Edition. (See figure 1. Lab setup for the exact specifications.)

All machines, operating systems, Kartris, and Application Defender were run using default configurations.

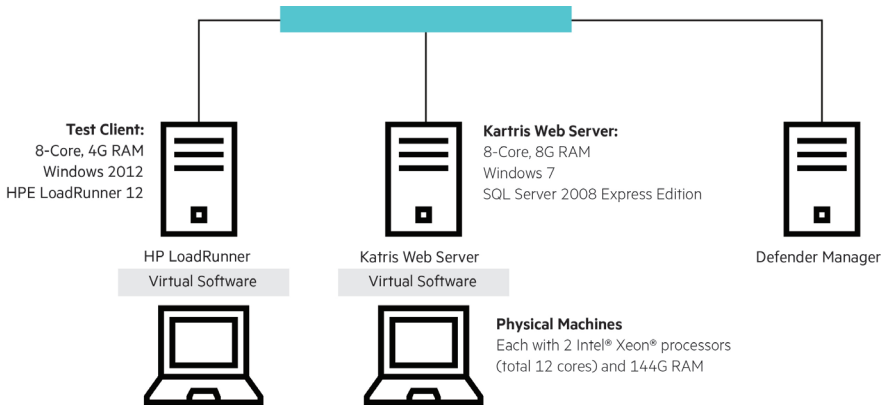


Figure 1. Lab setup

Testing methodology

We used HPE LoadRunner to measure the overhead added by the Application Defender agent (.NET). We created an HPE LoadRunner TruClient (Firefox) script with 95[1] steps, which included activities such as casual browsing, product searching, user registration, product purchasing, and admin functions. This represented one iteration. We executed this with 10 virtual users, each of which executed 30 iterations. An extra iteration per each virtual user was used to warm up the system to serve for tasks such as class loading, ASPX compilation, instrumentation, and so on.

The first warm-up iteration was noticeably slower than all the other iterations. There was a total of 95x10x30 = 28,500 steps. We first ran one complete set of tests against the raw application. The total server execution time (t1) was the time span between the HTTP request sent and the HTTP response received. We next ran the same set of tests against the application with the Application Defender agent installed and recorded the total server execution time (t2). By comparing t1 and t2, we determined the percentage of increase in execution time. We repeated this ten times, removed the highest and lowest numbers, and calculated the average. (See figure 2. Number of loops executed.)

[1] Due to static images, redirects, retries, and Ajax, there are around 188 HTTP request per iteration.

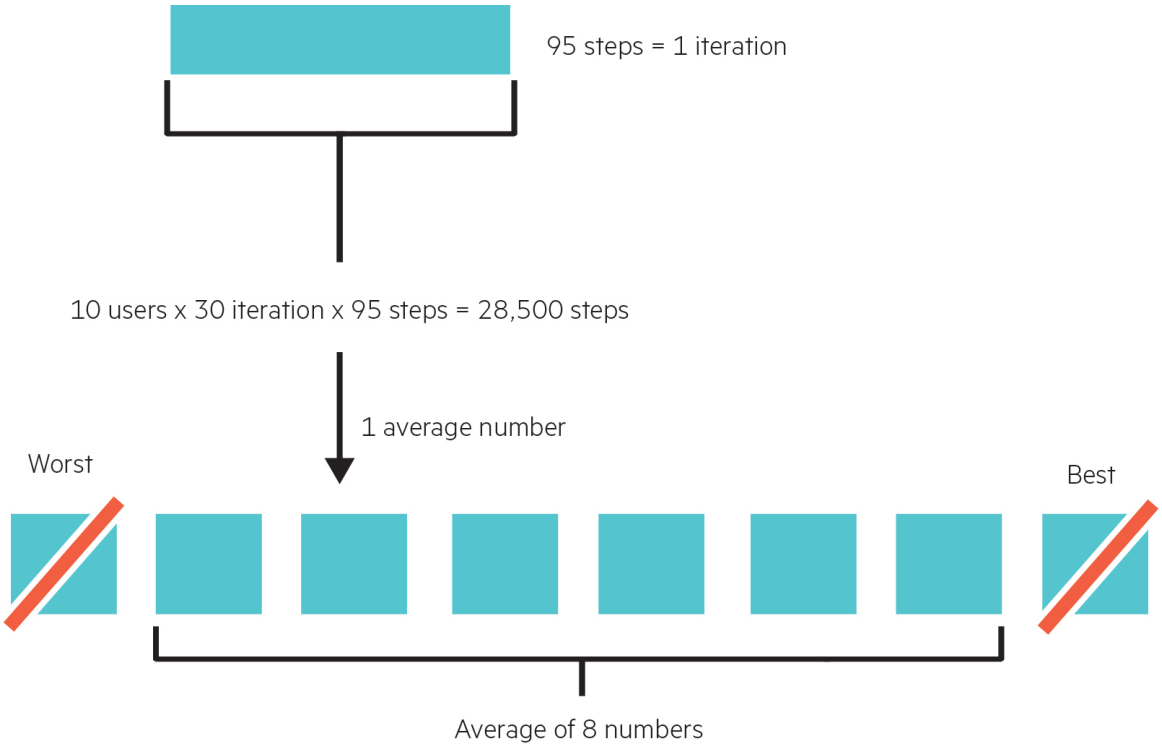


Figure 2. Number of loops executed

To test the performance of Application Defender while under attack, we also ran the same set of the tests with five extra attacks in each iteration. To measure the system performance under attack, we generated 100 steps in each iteration, which simulated 5 percent attack traffic. This part of the test was not designed to test the performance of the application protection rules; those rules are executed even on normal traffic. This test measures performance degradation caused by event generation and delivery.

To measure memory consumption, we used Windows® Performance Monitor to capture the total committed bytes.

Results

Increase in execution time

Protection rulepack only under normal traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10Threads30Loops1:	124.9417022	123.0785357	Error
10Threads30Loops2:	115.4727013	122.9030632	1.064347347
10Threads30Loops3:	113.1306637	122.2611717	1.080707633
10Threads30Loops4:	115.7629126	124.4206813	1.074788795
10Threads30Loops5:	125.5308303	129.1161284	1.028561096
10Threads30Loops6:	123.6637588	124.0063219	~~1.002770117~~ (min)
10Threads30Loops7:	112.5670539	126.0580824	~~1.119848819~~ (max)
10Threads30Loops8:	114.5372845	123.8387941	1.081209447
10Threads30Loops9:	114.6240919	122.7808285	1.071160751
10Threads30Loops10:	114.0507419	124.7781877	1.094058536
		Average (of 7)	**7.07%**

Protection rulepack only under 5% attack traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10Threads30Loops1:	113.7542765	123.9072957	1.089253956
10Threads30Loops2:	115.8914989	122.6565611	~~1.058374102~~ (min)
10Threads30Loops3:	115.1867907	124.6651087	1.082286502
10Threads30Loops4:	116.1024469	124.3251659	1.07082296
10Threads30Loops5:	113.5930051	126.9418701	~~1.117514851~~ (max)
10Threads30Loops6:	113.3001691	125.6196446	1.108733072
10Threads30Loops7:	117.7090105	125.1110574	1.062884285
10Threads30Loops8:	114.4058202	124.192271	1.085541547
10Threads30Loops9:	112.0001581	124.4161503	1.110856917
10Threads30Loops10:	112.539798	122.896451	1.092026583
		Average (of 8)	**8.78%**

Logging rulepack only under normal traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10Threads30Loops1:	117.880511	131.5177266	~~1.115686771~~ (min)
10Threads30Loops2:	130.5608883	146.4342914	1.121578547
10Threads30Loops3:	113.0832607	132.3803464	1.170644936
10Threads30Loops4:	123.366614	147.0043723	~~1.191605796~~ (max)
10Threads30Loops5:	129.0985601	153.4339278	1.188502239
10Threads30Loops6:	129.4987454	146.9958488	1.135114077
10Threads30Loops7:	127.0605606	148.9717437	1.172446769
10Threads30Loops8:	130.0146173	149.411039	1.149186469
10Threads30Loops9:	132.6233536	151.8902334	1.145275167
10Threads30Loops10:	126.8862026	148.5116393	1.170431743
		Average (of 8)	**15.66%**

Logging rulepack only under 5% attack traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10Threads30Loops1:	129.2428856	151.8635125	1.175024156
10Threads30Loops2:	129.3105833	148.0661361	1.145042674
10Threads30Loops3:	131.75781	151.4840866	1.149716184
10Threads30Loops4:	130.7567993	149.0517346	1.139915747
10Threads30Loops5:	131.2273814	150.8944398	1.149870082
10Threads30Loops6:	128.643241	150.5696478	1.17044352
10Threads30Loops7:	128.502891	145.2642516	1.130435669
10Threads30Loops8:	129.8105967	146.6994763	~~1.130104014~~ (min)
10Threads30Loops9:	131.5100085	154.6898028	~~1.176258785~~ (max)
10Threads30Loops10:	129.337659	148.5597655	1.148619564
		Average (of 8)	**15.11%**

Protection and logging under normal traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10Threads30Loops1:	114.7100737	136.2393889	1.187684608
10Threads30Loops2:	119.9705972	138.0094873	1.150360927
10Threads30Loops3:	116.7952383	138.5169803	1.185981401
10Threads30Loops4:	116.6452915	140.0141236	~~1.200340981~~ (max)
10Threads30Loops5:	116.8662106	135.0448218	1.155550617
10Threads30Loops6:	118.3872241	135.6576286	~~1.145880645~~ (min)
10Threads30Loops7:	118.7000771	138.9867694	1.170907154
10Threads30Loops8:	118.408257	138.8437022	1.17258463
10Threads30Loops9:	116.1820934	135.4612134	1.165938825
10Threads30Loops10:	117.9864766	137.240768	1.163190663
		Average (of 8)	**16.9%**

Protection and logging under 5% attack traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10Threads30Loops1:	115.8127396	138.0159044	1.191716083
10Threads30Loops2:	113.8611464	137.8647611	1.210814799
10Threads30Loops3:	117.9401584	140.059514	1.187547277
10Threads30Loops4:	118.1291655	138.3211574	~~1.17093147~~ (min)
10Threads30Loops5:	114.9392744	136.8238543	1.190401235
10Threads30Loops6:	116.839517	137.7957171	1.179358839
10Threads30Loops7:	115.8166057	140.2934019	1.211340991
10Threads30Loops8:	114.8130206	139.7620291	~~1.217301212~~ (max)
10Threads30Loops9:	115.6958309	140.2394131	1.212138865
10Threads30Loops10:	114.9273803	136.6659778	1.189150727
		Average (of 8)	**19.65%**

Increase in memory

Protection rulepack only under normal traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10VUsers30Iterations1	N/A	N/A	N/A
10VUsers30Iterations2	N/A	556	N/A
10VUsers30Iterations3	428	571	143
10VUsers30Iterations4	430	562	132
10VUsers30Iterations5	430	560	~~130~~ (min)
10VUsers30Iterations6	424	573	149
10VUsers30Iterations7	420	560	140
10VUsers30Iterations8	437	581	144
10VUsers30Iterations9	432	591	~~159~~ (max)
10VUsers30Iterations10	409	560	151
		Average (of 6)	**143**

Protection rulepack only under 5% attack traffic	Without App. Defender (unit: ms)	With App. Defender (unit: ms)	Increase
10VUsers30Iterations1	452	564	~~112~~ (min)
10VUsers30Iterations2	427	562	135
10VUsers30Iterations3	432	572	140
10VUsers30Iterations4	439	564	125
10VUsers30Iterations5	432	566	134
10VUsers30Iterations6	431	565	134
10VUsers30Iterations7	447	578	131
10VUsers30Iterations8	418	576	~~158~~ (max)
10VUsers30Iterations9	449	582	133
10VUsers30Iterations10	441	572	131
		Average (of 8)	**133**

Logging rulepack only under normal traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10VUsers30Iterations1	441	535	~~94~~ (min)
10VUsers30Iterations2	418	540	~~122~~ (max)
10VUsers30Iterations3	435	539	104
10VUsers30Iterations4	430	528	98
10VUsers30Iterations5	448	545	97
10VUsers30Iterations6	434	542	108
10VUsers30Iterations7	435	534	99
10VUsers30Iterations8	446	535	89
10VUsers30Iterations9	441	542	101
10VUsers30Iterations10	432	535	103
		Average (of 8)	**100**

Logging rulepack only under 5% attack traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10VUsers30Iterations1	428	547	~~119~~ (max)
10VUsers30Iterations2	458	536	78
10VUsers30Iterations3	475	538	~~63~~ (min)
10VUsers30Iterations4	429	544	115
10VUsers30Iterations5	446	538	92
10VUsers30Iterations6	449	541	92
10VUsers30Iterations7	420	539	119
10VUsers30Iterations8	443	536	93
10VUsers30Iterations9	428	544	116
10VUsers30Iterations10	446	544	98
		Average (of 8)	**100**

Protection and logging under normal traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10VUsers30Iterations1	405	556	151
10VUsers30Iterations2	419	555	136
10VUsers30Iterations3	450	562	112
10VUsers30Iterations4	436	563	127
10VUsers30Iterations5	426	560	134
10VUsers30Iterations6	422	563	141
10VUsers30Iterations7	429	568	139
10VUsers30Iterations8	409	568	~~159~~ (max)
10VUsers30Iterations9	460	556	~~96~~ (min)
10VUsers30Iterations10	439	571	132
		Average (of 8)	**134**

Protection and logging under 5% attack traffic	**Without App. Defender (unit: ms)**	**With App. Defender (unit: ms)**	**Increase**
10VUsers30Iterations1	439	580	141
10VUsers30Iterations2	426	571	145
10VUsers30Iterations3	463	577	114
10VUsers30Iterations4	437	562	125
10VUsers30Iterations5	454	576	122
10VUsers30Iterations6	440	571	131
10VUsers30Iterations7	441	569	128
10VUsers30Iterations8	480	572	~~92~~ (min)
10VUsers30Iterations9	422	577	~~155~~ (max)
10VUsers30Iterations10	448	584	136
		Average (of 8)	**130**

Summary and conclusion

Under normal traffic	Execution time overhead	Memory overhead
Protection only	7%	143
Logging only	16%	100
Protection and logging	17%	134

Under 5% attack traffic	Execution time overhead	Memory overhead
Protection only	9%	133
Logging only	15%	100
Protection and logging	20%	130

The logging rulepack generates around 4779 events per loop which is about 19 events per page, with the majority (~82%) of them being "Database Query" events. The rules are configurable granularly. You can disable some of the logging rules that don't address required use cases to generate fewer events and improve the overall performance. The protection rulepack performs more tasks when the application is under attack and this was reflected in the extra 2–3 percent degradation when the application was under 5% attack traffic.

Memory overhead for the logging only rulepack is likely lower due to fewer monitor classes in the logging rules. The overhead for "with" and "without" attack traffic is almost the same. This suggests the overhead is mainly due to extra libraries and classes loaded by the agent; the short-lived objects created during the analysis do not have any long-term impact. As a result, we expect the memory overhead to be effectively constant without much variation for different applications and times.

Learn more at

hpe.com/software/appdefender

Sign up for updates

4AA5-9269ENW, September 2016, Rev. 2

Hewlett Packard Enterprise