დანართი N2

მოთხოვნები ვებ აპლიკაციების და DNSSEC სისტემის მიმართ

ტექნიკური მოთხოვნები

1.) ორი ცალი ვირტუალურ აპლიანსი, შემდეგი ტექნიკური მახასიათებლებით:

1. სისტემის უნდა გააჩნდეს VMware vSphere ვირტუალიზაციის პლატფორმების მხარდაჭერა.
2. ვირტუალური აპლინასის ლიცენზია არ უნდა იყოს შეზღუდული ქსელური ნაკადის მიხედვით.
3. ვირტუალური აპლიანსის ლიცენზიას მხარდაჭერილი უნდა ჰქონდეს არანაკლებ 12 ბირთვის (vCPU) გამოყენების უფლება.
4. ვირტუალური აპლიანსის წარმადობის გაძლიერების მიზნით შესაძლებელი უნდა იყოს ბირთვების (vCPU) დამატება
5. ვირტუალურ აპლიანსს უნდა შეეძლოს როგროც Active/Acitve, ისე Active/Passive მაღამლდგრადობის რეჟიმში მუშაობა. აღნიშნულ რეჟიმში შესაძლებელი უნდა იყოს მინიმუმ 8 მოწყობილობის გაერთიანება, ასევე შესაძლებელი უნდა იყოს მათ შორის SSL და TCP სესიების სინქრონიზაცია.
6. ვირტუალური აპლინასის ლიცენზიას უნდა ჰქონდეს შემდეგი ფუნქციონალი: ვებ აპლიკაციების ფაიერვოლი (Web Application Firewall), HTTP ნაკადის ბალანსირება (HTTP Load Balancing), DNSSEC და GSLB (Global Server Load Balancing).
7. ქსელში გამართვის შემდეგი რეჟიმების მხარდაჭერა:
    7.1 L2 Forwarding
    7.2 L3 IP Forwarding (Routing)
    7.3 Packet Based Layer 4 traffic procesing
    7.4 Reverse Proxy and Forward Proxy (Layer 7)
8. შემდეგი ქსელური ტექნოლოგიების მხარდაჭერა:
    8.1 L2 კომუტაცია: Vlan, Vlan Groups, VXLAN.
    8.2 L3 მარშუტიზაცია: IPV4, IPV6, NAT, SNAT, ასევე სტატიკური მარშუტიზაციის მხარდაჭერა

2.) ვებ აპლიკაციის ფაირვოლი (Web Application Firewall) - დეტალური ტექნიკური მოთხოვნები ფუნქციონალის მიმართ

1.1 სისტემას უნდა შეეძლოს ვებ დონის აპლიკაციების DoS/DDoS შეტევებისგან დაცვა
1.2 სისტემას უნდა შეეძლოს იდენტიფიცირება შეტევად რესურსზე მიმართული დატვირთვის შესამცირებლად გაფილტროს ტრაფიკი და მოახდინოს მავნე ტრაფიკის ბლოკირება ვებ აპლიკაციისათვის
1.3 სისტემას უნდა შეეძლოს HTTP ტრაფიკის ფილტრაცია (HTTP და HTTPS პროტოკოლებზე დაფუძნებული შეტევა)
1.4 სისტემას უნდა შეეძლოს მავნე ტრაფიკის იდენტიფიცირება შემდეგი ფილტრაციის მექანიზმების გამოყენებით:
1.4.1 მოთხოვნების ფილტრაცია IP მისამართების რეპუტაციის მონაცემთა ბაზით:
1.4.1.1 IP მისამართების რეპუტაციის მონაცემთა ბაზის მართვის მექანიზმით უზრუნველყოფა.
1.4.1.2 IP მისამართების blacklist და whitelist შექმნის მექანიზმი
1.4.1.3 blacklist IP მისამართების კატეგორიზაციის მექანიზმი
1.4.1.4 სხვადასხვა კატეგორიის IP მისამართების blacklisting-სთვის სხვადასხვა ტრაფიკის დამუშავების წესების კონფიგურაციის მექანიზმი
1.4.2 OSI მოდელის 7-ე დონეზე DoS/DDoS-გან აპლიკაციების დაცვა:
1.4.3 სისტემას უნდა ქონდეს DDoS-სგან დაცვის კონფიგურაციის ვიზარდი
1.4.3.1 მავნე ტრაფიკის ბლოკირება შემდეგი კრიტერიუმებით:
1.4.3.1.1 Source IP მისამართი
1.4.3.1.2 მოწყობილობის ID
1.4.3.1.3 URL
1.4.2.1.4 ავტომატურად შექმნილი სიგნატურები
1.4.3.2 DDoS შეტევის ტრაფიკის capture
1.4.3.3 შეტევის ბლოკირების მახასიათებლები:
1.4.3.3.1 შეზღუდვა
1.4.3.3.2 ბლოკირება
1.4.3.3.3 captcha
1.4.4 გეოგრაფიული მდებარეობით ფილტრაცია (source ტრაფიკის ლოკაციით) როგორც გარკვეული რეგიონების გამორიცხვით ტრაფიკის შეზღუდვა, ასევე ტრაფიკის მიღება მხოლოდ გარკვეული რეგიონების სიიდან
სისტემას შეუძლია შემდეგი ოპერაციები: „track session Ids“, „prevent cookie injection, „cookie tampering“ და „session hijacking attacks” შეტევების აღმოჩენა და ბლოკირება
1.5 სისტემას უნდა ქონდეს ვებ აპლიკაციების დაცვის კონფიგურაციის ვიზარდი
1.6 სისტემას უნდა შეეძლოს რამდენიმე ვებ აპლიკაციის დაცვა SNI ან/და FQDN-ით ინდივიდუალურად თითოეული აპლიკაციისთვის სხვადასხვა უსაფრთხოების

პოლიტიკების გამოყენებით
1.7 სისტემას უნდა შეეძლოს “reverse proxy” რეჟიმში მუშაობისას განახორციელოს შემდეგი ოპერაციები: „digitally sign cookies“, „encrypt cookies“, „rewrite URLs“ და “rewrite response URLs”
1.8 სისტემას უნდა შეეძლოს შემდეგი ოპერაციები: „track session Ids“, „prevent cookie injection, „cookie tampering“ და „session hijacking attacks” შეტევების აღმოჩენა და ბლოკირება
1.9 სისტემას უნდა შეეძლოს ტრაფიკის ბლოკირება შემდეგი მეთოდებით:
- 1.9.1 Block the HTTP request
- 1.9.2 Block the connection
- 1.9.3 Block the IP address
- 1.9.4 Block the application session
- 1.9.5 Block the user
- 1.9.6 Send a TCP connection reset (in monitor mode)
- 1.9.7 Block the connection (in inline mode)

1.10 სისტემას უნდა შეეძლოს უზრუნველყოს XML კონტენტის უსაფრთხოება და მთლიანობა მათი სქემების მიხედვით (ისევე, როგორც SOAP, WSDL, JSON, AJAX)
1.11 სისტემას უნდა ქონდეს API უსაფრთხოების კონფიგურაციის ვიზარდი
1.12 სისტემას უნდა შეეძლოს swagger ფაილზე დაფუძნებით API უსაფრთხოების პოლიტიკის შექმნა
1.13 სისტემას უნდა ქონდეს უსაფრთხოების წესები WebSocket-სთვის
1.14 სისტემას უნდა ქონდეს GraphQL API-ს მხარდაჭერა
1.15 სისტემას უნდა ქონდეს ჩაშენებული უსაფრთხოების პოლიტიკები, რომლის საშუალებითაც განხორციელდება შესაბამისი საფრთხეების მოგერიება. სისტემას უნდა ქონდეს შემდეგი ჩაშენებული უსაფრთხოების პოლიტიკების მხარდაჭერა:
- 1.15.1 Apache Expect Header XSS
- 1.15.2 Cross-Site Request Forgery
- 1.15.3 Data Leakage Detection/Prevention
- 1.15.4 Directory Browsing Detection
- 1.15.5 Directory Traversal (In Cookies/Parameters Value)
- 1.15.6 Fullwidth/Halfwidth Unicode Decoding
- 1.15.7 HTTP Response Splitting Vulnerability
- 1.15.8 IE Discussion BarAccess to Internal Information
- 1.15.9 Malformed HTTP Attack (Non-compatible HTTP Results Error code)
- 1.15.10 OS Command Injection
- 1.15.11 Plain Vanilla Scanner Detection
- 1.15.12 Privacy Violation -Credit Card Number Insertion
- 1.15.13 Privacy Violation -Credit Card Number Insertion by Internal IP Address

1.15.14 Privacy Violation -Credit Card Number Insertion by non-Internal IP Address

1.15.15 Sensitive Error Messages Leakage

1.15.16 Suspected parameter tampering -Deprecated

1.15.17 Suspicious Response Code

1.15.18 Webdav Method Detections

1.16 სისტემას უნდა შეეძლოს შემდეგი ტიპის ვებ აპლიკაციაზე მიმდინარე შეტევების აღმოჩენა და თავიდან აცილება:

1.16.1 OWASP Top 10

1.16.2 OS Command Injection

1.16.3 SQL Injection

1.16.4 Session Hijacking

1.16.5 Site Reconnaissance

1.16.6 Site Scraping

1.16.7 Cross-Site Scripting

1.16.8 Cross-Site Request Forgery

1.17 სისტემას უნდა შეეძლოს Zero-Day Web Worm შეტევებისგან დაცვა

1.18 სისტემის უნდა შეეძლოს ინტეგრაცია WhiteHat, IBM, Cenzic და HP Qualys ვებ სისუსტეების გამომვლენ სკანერებთან და აღნიშნული სკანერების მიერ დაგენერირებული რეპორტის მიხედვით პოლიტიკების ავტომატური შექმნა. ასევე სისტემას უნდა შეეძლოს ე.წ „Application Virtual Patching“ განხორციელება

1.19 სისტემას უნდა შეეძლოს აპლიკაციების უსაფრთხოების პოლიტიკების შექმნა შემდეგი მეთოდებით:

1.19.1 რეალური აპლიკაციის ტრაფიკის საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა (self-learning mode)

1.19.2 სისუსტეების გამომვლენი სკანერის რეპორტის საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა

1.19.2 წინასწარ განსაზღვრული შაბლონების საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა

1.19.3 არსებული წესების fine-tuning მექანიკურ რეჟიმში

1.19.4 გამოყოფილი აპლიკაციებისთვის უსაფრთხოების წესების მექანიკურ რეჟიმში შექმნა

1.19.5 სისტემას უნდა შეეძლოს BotNet-ებისგან დაცვა:

1.19.5.1 სისტემას უნდა ქონდეს ბოტებისაგან დაცვის კონფიგურაციის ვიზარდი

1.19.5.2 სისტემას უნდა ქონდეს BotNet კატეგორიზაცის ბაზა

1.19.5.3 სისტემას უნდა შეეძლოს ინდივიდუალური წესების დამუშავება თითოეული კატეგორიისათვის

1.19.5.4 სისტემას უნდა შეეძლოს ჩაშენებული JavaScript-ის ან ქცევაზე

დაფუძნებული ანალიტიკის (behavioral analysis) ავტომატიზირებული მექანიზმით BotNet-ის აღმოჩენა

1. 19.5.5 სისტემას უნდა ქონდეს CAPTCHA BotNet-ისაგან დაცვის მექანიზმი
1. 19.5.6 სისტემას უნდა ქონდეს page stub for bots past CAPTCHA
1. 19.5.7 სისტემას უნდა შეეძლოს ქცევაზე დაფუძნებული ანალიტიკით ბოტების ავტომატური აღმოჩენა

1.20 სისტემას უნდა ქონდეს App უსაფრთხოების პოლიტიკების შემდეგი მეთოდებით:

1.20.1 რეალური ტრაფიკის ქცევაზე დაფუძნებული ანალიტიკის საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა.

1.20.2 სისუსტეების გამომვლენი სკანერის რეპორტის საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა.

1.20.3 რეალური App ტრაფიკის საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა.

1.20.4 წინასწარ განსაზღვრული შაბლონების საფუძველზე უსაფრთხოების წესების ავტომატური შექმნა

1.20.5 არსებული წესების fine-tuning მექანიკურ რეჟიმში

1.20.6 სპეციფიური აპლიკაციებისთვის შესაბამისი უსაფრთხოების წესების შექმნა ავტომატურ და მექანიკურ რეჟიმში

1.21 სისტემას უნდა შეეძლოს გააგზავნოს webhook მოვლენები

1.22 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით აპლიკაციისათვის საფრთხის ხარისხის განსაზღვრა

1.23 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით საფრთხის დეტალური აღწერა

1.24 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით დანაკარგის დეტალური აღწერა დაცული აპლიკაციისთვის

1.25 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით უსაფრთხოების პოლიტიკის ცვლილებებზე რეკომენდაციებით უზრუნველყოფა

1.26 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით დაუმუშავებელი ტრაფიკის capture შეტევის დროს

1.27 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით ტრაფიკის მონიტორინგი დაცული რესურსებისთვის ანომალიების გამოსავლენად

1.28 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით აღმოჩენილ ანომალიებზე გაფრთხილების გაგზავნა

1.29 სისტემას უნდა შეეძლოს აპლიკაციის უსაფრთხოების პოლიტიკის თითოეული მოვლენის შეფასებით შეტევების იდენტიფიცირება და აღკვეთა, რომლებიც

ასკანირებენ ვებ გვერდების სისუსტეებს
1.30 სისტემას უნდა შეეძლოს აპლიკაციის სერვერზე გამოყენებული ტექნოლოგიების (OS, web server, web application technology) ავტომატური განსაზღვრა
1.31 სისტემას უნდა შეეძლოს მომხმარებლის მხარეს აპლიკაციის მონაცემთა ველის მნიშვნელობების არასავალდებულო დაწშიფვრა
1.32 სისტემას უნდა შეეძლოს ანტივირუსთან ICAP-ით ინტეგრაცია
1.33 სისტემას უნდა შეეძლოს CAPTCHA Auto Bypass-ის განსაზღვრა
1.34 სისტემას უნდა შეეძლოს WAF Security Bypass მცდელობების განსაზღვრა
1.35 სისტემას უნდა შეეძლოს კლიენტის fingerprint-ის შექმნა თვალთვალისათვის, იმ შემთხვევაშიც კი, როდესაც ერთი მომხმარებელი ცდილობს გამოიყენოს რამდენიმე სესია.
1.36 სისტემას უნდა შეეძლოს დაინფიცირებული კლიენტების fingerprint-მონაცემთა ბაზის საფუძველზე მათი დაბლოკვა
1.37 სისტემას უნდა ქონდეს მხარდაჭერა და შეეძლოს გამოყენება წინასწარ განსაზღვრული პოლიტიკების და რეპორტების, მათ შორის PCI DSS აუდიტისთვის
1.38 სისტემას უნდა შეეძლოს სიგნატურების შეცვლა ან დამატება.
1.39 სისტემას უნდა ქონდეს მიმდინარე შეტევებზე დაყრდნობით მწარმოებლის SoC -დან დამატებითი სიგნატურები
1.40 სისტემას უნდა ქონდეს მხარდაჭერა შექმნას სისუსტეების ტიპები, მათი განმარტების წესები, აღწეროს და მოხდეს ასახვა შიდა და უსაფრთხოების მოვლენების მონიტორინგის სისტემაში
1.41 სისტემას უნდა შეეძლოს health მონიტორინგი დაცული აპლიკაციის და მოახდინოს ტრაფიკის გადართვა სხვა სერვერზე ძირითადი აპლიკაციის გაუმართაობის შემთხვევაში
1.42 სისტემას უნდა შეეძლოს დაბლოკილ ან საეჭვო მოთხოვნაზე უსაფრთხოების პოლიტიკაში აუცილებელი გამონაკლისების დაშვება
143 სისტემას უნდა გააჩნდეს მიმდინარე რისკების კორელაციის შესაძლებლობა. აღნიშნული ფუნქციის საშუალებით შესაძლებელი უნდა იყოს კომპლექსური, რამოდენიმე ეტაპად განხორციელებული შეტევების ერთმანეთან კორელაცია და მოგერიება.

3.) საკვალიფიკაციო მოთხოვნები მომწოდებლის და მწარმოებლის მიმართ

1) პრეტენდენტმა უნდა წარმოადგინოს შემოთავაზებული მწარმოებლის საქართველოს ტერიტორიაზე ოფიციალური პარტნიორობის დამადასტურებელი დოკუმენტი.

2) პრეტენდენტმა უნდა წარმოადგინოს მწარმოებლის ავტორიზაცია (MAF), რომელიც უფლებას აძლევს პრეტენდენტს აღნიშნული პროექტის ფარგლებში საქართველოს ტერიტორიაზე განახორციელოს პროდუქციის გაყიდვა და სერვისული მომსახურება.

3) პრეტენდენტმა უნდა წარმოადგინოს პრეტენდენტ კომპანიაში დასაქმებული მინიმუმ ერთი თანამშრომლის სერტიფიკატი, რომელსაც გააჩნია შემოთავაზებულ სისტემაზე მწარმოებლის გაცემული ტექნიკური სერტიკიფატი.

4) პრეტენდენტმა უნდა წარმოადგინოს ინფორმაცია ბოლო 3 (სამი) წლის განმავლობაში განხორციელებული ანალოგიური მწარმოებლის 3 (სამი) სხვადასხვა პროექტის შესახებ

4.) მოთხოვნები საგარანტიო და სერვისულ მომსახურეობაზე

გადაწყვეტილებაზე უნდა ვრცელდებოდეს პროგრამული უზრუნველყოფის და სხვა აუცილებელი კომპონენტების მინიმუმ 1 წლიანი განახლებების სერვისი.