United States Patent 12395488
Kind Code B2
Date of Patent August 19, 2025
Inventor(s) Lidgi; Matilda et al.

Techniques for analyzing external exposure in cloud environments

Abstract

A system and method for performing active inspection of a cloud computing environment includes receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from a network which is external to the cloud computing environment; and actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from a network external to the cloud computing environment.

Inventors: **Lidgi; Matilda (Tel Aviv, IL), Keren; Shai (Tel Aviv, IL), Herzberg; Raaz (Tel Aviv, IL), Lichtenstein; Avi Tal (Tel Aviv, IL), Luttwak; Ami (Binyamina, IL), Reznik; Roy (Tel Aviv, IL)**

Applicant: **Wiz, Inc.** (New York, NY)

Family ID: **1000008763361**

Assignee: **Wiz, Inc. (New York, NY)**

Appl. No.: **17/659165**

Filed: **April 13, 2022**

Prior Publication Data

Document Identifier	Publication Date
US 20230336554 A1	Oct. 19, 2023

Publication Classification

Int. Cl.: **H04L29/00** (20060101); **H04L9/40** (20220101); **H04L61/5007** (20220101)

U.S. Cl.:

CPC	**H04L63/101** (20130101); **H04L61/5007** (20220501); **H04L63/102** (20130101); **H04L63/205** (20130101);

Field of Classification Search

CPC:	H04L (63/101); H04L (61/5007); H04L (63/102); H04L (63/205); H04L (63/1433); H04L (63/10); H04L (63/20); H04L (63/143)

References Cited

U.S. PATENT DOCUMENTS

Patent No.	Issued Date	Patentee Name	U.S. Cl.	CPC
6910132	12/2004	Bhattacharya	N/A	N/A
7627652	12/2008	Commons et al.	N/A	N/A
7784101	12/2009	Verbowski et al.	N/A	N/A
8104075	12/2011	Spector	N/A	N/A
8200965	12/2011	Fujibayashi et al.	N/A	N/A
8320558	12/2011	Zea	N/A	N/A
8352431	12/2012	Protopopov et al.	N/A	N/A
8412688	12/2012	Armangau et al.	N/A	N/A
8413239	12/2012	Sutton	726/23	H04L 63/145
8417967	12/2012	Foster et al.	N/A	N/A
8499354	12/2012	Satish et al.	N/A	N/A
8595822	12/2012	Schrecker et al.	N/A	N/A
8701200	12/2013	Naldurg et al.	N/A	N/A
8789049	12/2013	Hutchins et al.	N/A	N/A
8813234	12/2013	Bowers et al.	N/A	N/A
8898481	12/2013	Osburn, III et al.	N/A	N/A
8904525	12/2013	Hodgman et al.	N/A	N/A
8914406	12/2013	Haugsnes	710/316	G06F 9/45558
9009836	12/2014	Yarykin et al.	N/A	N/A
9094379	12/2014	Miller	N/A	N/A
9119017	12/2014	Sinha	N/A	H04W 12/128
9165142	12/2014	Sanders et al.	N/A	N/A
9172621	12/2014	Dippenaar	N/A	N/A
9185136	12/2014	Dulkin et al.	N/A	N/A
9330273	12/2015	Khetawat et al.	N/A	N/A
9369433	12/2015	Paul	N/A	H04L 63/029
9419996	12/2015	Porat	N/A	G06F 21/566
9438634	12/2015	Ross et al.	N/A	N/A
9467473	12/2015	Jayaraman	N/A	H04L 63/20
9544327	12/2016	Sharma et al.	N/A	N/A
9563385	12/2016	Kowalski et al.	N/A	N/A
9569328	12/2016	Pavlov et al.	N/A	N/A
9582662	12/2016	Messick et al.	N/A	N/A
9596235	12/2016	Badam et al.	N/A	N/A
9607104	12/2016	Turner et al.	N/A	N/A
9621595	12/2016	Lian et al.	N/A	N/A

9646172	12/2016	Hahn	N/A	N/A
9661009	12/2016	Karandikar et al.	N/A	N/A
9665465	12/2016	Jain et al.	N/A	N/A
9672355	12/2016	Titonis et al.	N/A	N/A
9712503	12/2016	Ahmed	N/A	N/A
9892261	12/2017	Joram et al.	N/A	N/A
9992186	12/2017	Drozd et al.	N/A	N/A
10002247	12/2017	Suarez et al.	N/A	N/A
10009337	12/2017	Fischer et al.	N/A	N/A
10032032	12/2017	Suarez et al.	N/A	N/A
10044723	12/2017	Fischer et al.	N/A	N/A
10063445	12/2017	Preece	N/A	N/A
10135826	12/2017	Reddy	N/A	H04L 67/06
10205638	12/2018	Angrish et al.	N/A	N/A
10229125	12/2018	Goodman et al.	N/A	N/A
10255370	12/2018	Carpenter et al.	N/A	N/A
10360025	12/2018	Foskett et al.	N/A	N/A
10412103	12/2018	Haugsnes	N/A	N/A
10412109	12/2018	Loureiro et al.	N/A	N/A
10459664	12/2018	Dreier et al.	N/A	N/A
10503904	12/2018	Singh et al.	N/A	N/A
10509909	12/2018	Andriani	N/A	N/A
10536471	12/2019	Derbeko et al.	N/A	N/A
10540499	12/2019	Wailly et al.	N/A	N/A
10552610	12/2019	Vashisht et al.	N/A	N/A
10554507	12/2019	Siddiqui et al.	N/A	N/A
10567468	12/2019	Perlmutter	N/A	H04L 67/563
10572226	12/2019	Biskup et al.	N/A	N/A
10574675	12/2019	Peppe	N/A	G06F 21/577
10623386	12/2019	Bernat et al.	N/A	N/A
10630642	12/2019	Clark et al.	N/A	N/A
10664619	12/2019	Marelas	N/A	G06N 5/01
10691636	12/2019	Tabaaloute et al.	N/A	N/A
10721260	12/2019	Schlarp et al.	N/A	N/A
10725775	12/2019	Suarez et al.	N/A	N/A
10728252	12/2019	Desai et al.	N/A	N/A
10735430	12/2019	Stoler	N/A	N/A
10735442	12/2019	Swackhamer	N/A	G06F 21/56
10791138	12/2019	Siddiqui et al.	N/A	N/A
10803188	12/2019	Rajput et al.	N/A	N/A
10831898	12/2019	Wagner	N/A	N/A
10915626	12/2020	Tang	N/A	H04L 41/142
10924503	12/2020	Pereira et al.	N/A	N/A
10949406	12/2020	Calvo et al.	N/A	N/A
10972484	12/2020	Swackhamer	N/A	H04L 63/1416
10997293	12/2020	Wiest et al.	N/A	N/A
11005860	12/2020	Glyer et al.	N/A	N/A
11016954	12/2020	Babocichin et al.	N/A	N/A
11044118	12/2020	Reed et al.	N/A	N/A

11055414	12/2020	Claes	N/A	N/A
11064032	12/2020	Yang	N/A	H04L 45/74
11099976	12/2020	Khakare et al.	N/A	N/A
11102231	12/2020	Kraning et al.	N/A	N/A
11165652	12/2020	Byrne	N/A	H04L 43/0847
11216563	12/2021	Veselov et al.	N/A	N/A
11245730	12/2021	Bailey	N/A	N/A
11271961	12/2021	Berger	N/A	G06F 9/451
11334670	12/2021	Franco et al.	N/A	N/A
11336555	12/2021	Soh et al.	N/A	N/A
11366897	12/2021	Ramanathan et al.	N/A	N/A
11388183	12/2021	Hoopes et al.	N/A	N/A
11397808	12/2021	Prabhu et al.	N/A	N/A
11405426	12/2021	Nguyen	N/A	N/A
11418528	12/2021	Deardorff et al.	N/A	N/A
11442989	12/2021	Dvinov et al.	N/A	N/A
11444974	12/2021	Shakhzadyan	N/A	G06F 21/577
11483317	12/2021	Bolignano et al.	N/A	N/A
11496498	12/2021	Wright et al.	N/A	N/A
11496519	12/2021	Gupta et al.	N/A	N/A
11503063	12/2021	Rao	N/A	H04L 63/1416
11507672	12/2021	Pagnozzi et al.	N/A	N/A
11509658	12/2021	Kulkarni	N/A	N/A
11516222	12/2021	Srinivasan et al.	N/A	N/A
11520907	12/2021	Borowiec et al.	N/A	N/A
11546360	12/2022	Woodford et al.	N/A	N/A
11556659	12/2022	Kumar et al.	N/A	N/A
11558401	12/2022	Vashisht et al.	N/A	N/A
11558414	12/2022	Nguyen	N/A	N/A
11558423	12/2022	Gordon et al.	N/A	N/A
11567751	12/2022	Cosentino et al.	N/A	N/A
11570090	12/2022	Shen et al.	N/A	N/A
11575696	12/2022	Ithal	N/A	G06F 16/95
11606378	12/2022	Delpont et al.	N/A	N/A
11614956	12/2022	Tsirkin et al.	N/A	N/A
11645390	12/2022	Vijayvargiya et al.	N/A	N/A
11651055	12/2022	Regier et al.	N/A	N/A
11662928	12/2022	Kumar et al.	N/A	N/A
11663340	12/2022	Wu et al.	N/A	N/A
11669386	12/2022	Abrol	N/A	N/A
11695785	12/2022	Ithal et al.	N/A	N/A
11700233	12/2022	St. Pierre	N/A	N/A
11720685	12/2022	Gwilliams	N/A	N/A
11750566	12/2022	Montilla Lugo	N/A	N/A
11757844	12/2022	Xiao	726/11	H04L 63/0281
11770398	12/2022	Erlingsson	709/224	G06F 16/3329

11792284	12/2022	Nanduri	709/224	G06F 9/542
11799874	12/2022	Lichtenstein et al.	N/A	N/A
11803766	12/2022	Srinivasan	N/A	G06N 5/04
11831670	12/2022	Molls et al.	N/A	N/A
11841945	12/2022	Fogel	N/A	G06F 21/53
11902282	12/2023	Ghiold et al.	N/A	N/A
11914707	12/2023	Ramanathan et al.	N/A	N/A
11922220	12/2023	Haghighat et al.	N/A	N/A
11936785	12/2023	Shemesh et al.	N/A	N/A
11960609	12/2023	Gokhman et al.	N/A	N/A
11973770	12/2023	Miran et al.	N/A	N/A
11991216	12/2023	Venkatachari	N/A	N/A
12003541	12/2023	Shulman et al.	N/A	N/A
12019770	12/2023	Nilsson et al.	N/A	N/A
12050696	12/2023	Pieno et al.	N/A	N/A
12058177	12/2023	Crabtree et al.	N/A	N/A
12063305	12/2023	Ip et al.	N/A	N/A
12166785	12/2023	Yellapragada et al.	N/A	N/A
2002/0184486	12/2001	Kershenbaum et al.	N/A	N/A
2003/0188194	12/2002	Currie et al.	N/A	N/A
2003/0217039	12/2002	Kurtz et al.	N/A	N/A
2005/0050365	12/2004	Seki	726/4	H04L 63/1466
2005/0251863	12/2004	Sima	N/A	N/A
2005/0283645	12/2004	Turner et al.	N/A	N/A
2007/0174915	12/2006	Gribble et al.	N/A	N/A
2007/0271360	12/2006	Sahita et al.	N/A	N/A
2008/0075283	12/2007	Takahashi	N/A	N/A
2008/0221833	12/2007	Brown et al.	N/A	N/A
2008/0307020	12/2007	Ko et al.	N/A	N/A
2008/0320594	12/2007	Jiang	N/A	N/A
2009/0106256	12/2008	Safari et al.	N/A	N/A
2009/0271863	12/2008	Govindavajhala et al.	N/A	N/A
2010/0242082	12/2009	Keene et al.	N/A	N/A
2010/0263049	12/2009	Cross et al.	N/A	N/A
2010/0281275	12/2009	Lee et al.	N/A	N/A
2011/0035802	12/2010	Arajujo, Jr. et al.	N/A	N/A
2011/0055361	12/2010	Dehaan	N/A	N/A
2011/0276806	12/2010	Casper et al.	N/A	N/A
2012/0110651	12/2011	Van Biljon et al.	N/A	N/A
2012/0255003	12/2011	Sallam	N/A	N/A
2012/0297206	12/2011	Nord et al.	N/A	N/A
2012/0311696	12/2011	Datsenko et al.	N/A	N/A
2013/0024940	12/2012	Hutchins et al.	N/A	N/A
2013/0054890	12/2012	Desai et al.	N/A	N/A
2013/0124669	12/2012	Anderson et al.	N/A	N/A
2013/0160119	12/2012	Sartin	N/A	N/A
2013/0160129	12/2012	Sartin	N/A	N/A
2013/0290708	12/2012	Diaz et al.	N/A	N/A

2014/0096134	12/2013	Barak	N/A	N/A
2014/0115578	12/2013	Cooper	718/1	H04L 63/205
2014/0237537	12/2013	Manmohan	N/A	N/A
2014/0317677	12/2013	Vaidya	726/1	H04L 63/20
2014/0337613	12/2013	Martini	N/A	N/A
2015/0033305	12/2014	Shear	726/11	G06F 21/53
2015/0055647	12/2014	Roberts	370/352	H04L 65/1063
2015/0058993	12/2014	Choi et al.	N/A	N/A
2015/0095995	12/2014	Bhalerao	N/A	N/A
2015/0163192	12/2014	Jain	370/255	H04L 61/103
2015/0172321	12/2014	Kirti et al.	N/A	N/A
2015/0254364	12/2014	Piduri et al.	N/A	N/A
2015/0304302	12/2014	Zhang et al.	N/A	N/A
2015/0310215	12/2014	McBride et al.	N/A	N/A
2015/0319160	12/2014	Ferguson et al.	N/A	N/A
2016/0063466	12/2015	Sheridan et al.	N/A	N/A
2016/0078231	12/2015	Bach et al.	N/A	N/A
2016/0103669	12/2015	Gamage et al.	N/A	N/A
2016/0105454	12/2015	Li	726/23	H04L 63/1416
2016/0140352	12/2015	Nickolov	N/A	N/A
2016/0156664	12/2015	Nagaratnam	726/1	H04W 12/06
2016/0224600	12/2015	Munk	N/A	N/A
2016/0299708	12/2015	Yang et al.	N/A	N/A
2016/0366185	12/2015	Lee	N/A	H04L 63/20
2017/0026416	12/2016	Carpenter et al.	N/A	N/A
2017/0034198	12/2016	Powers et al.	N/A	N/A
2017/0070506	12/2016	Reddy	N/A	H04L 63/062
2017/0104755	12/2016	Arregoces	N/A	H04L 67/10
2017/0111384	12/2016	Loureiro et al.	N/A	N/A
2017/0163650	12/2016	Seigel et al.	N/A	N/A
2017/0180421	12/2016	Shieh et al.	N/A	N/A
2017/0185784	12/2016	Madou	N/A	H04L 63/1433
2017/0187686	12/2016	Shaikh et al.	N/A	N/A
2017/0187743	12/2016	Madou	N/A	G06F 21/566
2017/0200122	12/2016	Edson et al.	N/A	N/A
2017/0223024	12/2016	Desai	N/A	H04L 63/20
2017/0230179	12/2016	Mannan et al.	N/A	N/A
2017/0237560	12/2016	Mueller et al.	N/A	N/A
2017/0257347	12/2016	Yan	N/A	H04N 1/32352
2017/0285978	12/2016	Manasse	N/A	N/A
2017/0300690	12/2016	Ladnai et al.	N/A	N/A
2017/0374136	12/2016	Ringdahl	N/A	N/A
2018/0004950	12/2017	Gupta	N/A	G06F 21/52
2018/0007087	12/2017	Grady et al.	N/A	N/A
2018/0026995	12/2017	Dufour et al.	N/A	N/A

2018/0027009	12/2017	Santos	726/25	H04L 63/1441
2018/0063290	12/2017	Yang et al.	N/A	N/A
2018/0081640	12/2017	Collins	N/A	N/A
2018/0101622	12/2017	Helvik et al.	N/A	N/A
2018/0137174	12/2017	Cahana et al.	N/A	N/A
2018/0150412	12/2017	Manasse	N/A	N/A
2018/0159882	12/2017	Brill	N/A	N/A
2018/0181310	12/2017	Feinberg et al.	N/A	N/A
2018/0191726	12/2017	Luukkala	N/A	N/A
2018/0219888	12/2017	Apostolopoulos	N/A	N/A
2018/0234459	12/2017	Kung	N/A	H04L 63/0263
2018/0239902	12/2017	Godard	N/A	G06F 21/53
2018/0260566	12/2017	Chaganti et al.	N/A	N/A
2018/0270268	12/2017	Gorodissky et al.	N/A	N/A
2018/0276084	12/2017	Mitkar et al.	N/A	N/A
2018/0278639	12/2017	Bernstein et al.	N/A	N/A
2018/0288129	12/2017	Joshi et al.	N/A	N/A
2018/0307736	12/2017	Balakrishnan et al.	N/A	N/A
2018/0309747	12/2017	Sweet et al.	N/A	N/A
2018/0321993	12/2017	McClory	N/A	H04L 41/5041
2018/0341768	12/2017	Marshall et al.	N/A	N/A
2018/0349612	12/2017	Harel et al.	N/A	N/A
2018/0359058	12/2017	Kurian	N/A	H04L 63/0227
2018/0359059	12/2017	Kurian	N/A	H04L 63/1483
2018/0367548	12/2017	Stokes, III et al.	N/A	N/A
2019/0007271	12/2018	Rickards et al.	N/A	N/A
2019/0018961	12/2018	Kostyushko et al.	N/A	N/A
2019/0043201	12/2018	Strong et al.	N/A	N/A
2019/0058722	12/2018	Levin et al.	N/A	N/A
2019/0068617	12/2018	Coleman	N/A	H04L 63/0876
2019/0068627	12/2018	Thampy	N/A	N/A
2019/0081963	12/2018	Waghorn	N/A	N/A
2019/0089720	12/2018	Aditham et al.	N/A	N/A
2019/0104140	12/2018	Gordeychik et al.	N/A	N/A
2019/0116111	12/2018	Izard et al.	N/A	N/A
2019/0121986	12/2018	Stopel et al.	N/A	N/A
2019/0132350	12/2018	Smith et al.	N/A	N/A
2019/0149604	12/2018	Jahr	N/A	N/A
2019/0166129	12/2018	Gaetjen et al.	N/A	N/A
2019/0171811	12/2018	Daniel et al.	N/A	N/A
2019/0191417	12/2018	Baldemair et al.	N/A	N/A
2019/0205267	12/2018	Richey et al.	N/A	N/A
2019/0207966	12/2018	Vashisht et al.	N/A	N/A
2019/0220298	12/2018	Jiao et al.	N/A	N/A

2019/0220575	12/2018	Boudreau et al.	N/A	N/A
2019/0229915	12/2018	Digiambattista et al.	N/A	N/A
2019/0235900	12/2018	Singh et al.	N/A	N/A
2019/0236409	12/2018	Van Der Stockt et al.	N/A	N/A
2019/0245883	12/2018	Gorodissky et al.	N/A	N/A
2019/0260764	12/2018	Humphrey et al.	N/A	N/A
2019/0278928	12/2018	Rungta et al.	N/A	N/A
2019/0327271	12/2018	Saxena et al.	N/A	N/A
2019/0334715	12/2018	Gray	N/A	N/A
2019/0354675	12/2018	Gan et al.	N/A	N/A
2019/0377988	12/2018	Qi et al.	N/A	N/A
2020/0007314	12/2019	Vouk et al.	N/A	N/A
2020/0007569	12/2019	Dodge et al.	N/A	N/A
2020/0012659	12/2019	Dageville et al.	N/A	N/A
2020/0012818	12/2019	Levin et al.	N/A	N/A
2020/0028862	12/2019	Lin	N/A	H04L 63/104
2020/0044916	12/2019	Kaufman et al.	N/A	N/A
2020/0050440	12/2019	Chuppala et al.	N/A	N/A
2020/0074360	12/2019	Humphries et al.	N/A	N/A
2020/0082094	12/2019	McAllister et al.	N/A	N/A
2020/0106782	12/2019	Sion	N/A	G06F 21/31
2020/0117434	12/2019	Biskup et al.	N/A	N/A
2020/0125352	12/2019	Kannan	N/A	G06F 8/65
2020/0137097	12/2019	Zimmermann et al.	N/A	N/A
2020/0137125	12/2019	Patnala et al.	N/A	N/A
2020/0145405	12/2019	Bosch	N/A	H04L 45/308
2020/0186416	12/2019	Hashimoto et al.	N/A	N/A
2020/0244678	12/2019	Shua	N/A	N/A
2020/0244692	12/2019	Shua	N/A	N/A
2020/0259852	12/2019	Wolff et al.	N/A	N/A
2020/0287927	12/2019	Zadeh et al.	N/A	N/A
2020/0320189	12/2019	Zhang et al.	N/A	N/A
2020/0320845	12/2019	Livny et al.	N/A	N/A
2020/0336489	12/2019	Wuest et al.	N/A	N/A
2020/0382556	12/2019	Woolward et al.	N/A	N/A
2020/0387357	12/2019	Mathon et al.	N/A	N/A
2020/0389431	12/2019	St. Pierre	N/A	H04L 43/0876
2020/0389469	12/2019	Litichever	N/A	H04W 4/40
2020/0409741	12/2019	Dornemann et al.	N/A	N/A
2021/0014265	12/2020	Hadar et al.	N/A	N/A
2021/0026932	12/2020	Boudreau et al.	N/A	N/A
2021/0042263	12/2020	Zdornov et al.	N/A	N/A
2021/0056548	12/2020	Monica et al.	N/A	N/A
2021/0089662	12/2020	Muniswamy-Reddy et al.	N/A	N/A
2021/0105304	12/2020	Kraning	N/A	G06Q 10/08
2021/0144517	12/2020	Guim Bernat et al.	N/A	N/A

2021/0149788	12/2020	Downie	N/A	G06F 11/3604
2021/0158835	12/2020	Hill et al.	N/A	N/A
2021/0168150	12/2020	Ross et al.	N/A	N/A
2021/0173939	12/2020	Kotler et al.	N/A	N/A
2021/0176123	12/2020	Plamondon	N/A	H04L 41/0843
2021/0176164	12/2020	Kung	N/A	H04L 69/22
2021/0185073	12/2020	Ewaida et al.	N/A	N/A
2021/0194678	12/2020	Schindewolf et al.	N/A	N/A
2021/0200881	12/2020	Joshi et al.	N/A	N/A
2021/0203684	12/2020	Maor et al.	N/A	N/A
2021/0211453	12/2020	Cooney	N/A	G06F 16/958
2021/0216591	12/2020	Dvinov et al.	N/A	N/A
2021/0216630	12/2020	Karr	N/A	N/A
2021/0218567	12/2020	Richards et al.	N/A	N/A
2021/0226812	12/2020	Park	N/A	N/A
2021/0226928	12/2020	Crabtree et al.	N/A	N/A
2021/0232344	12/2020	Corrie	N/A	N/A
2021/0234889	12/2020	Burle et al.	N/A	N/A
2021/0263802	12/2020	Gottemukkula et al.	N/A	N/A
2021/0297447	12/2020	Crabtree et al.	N/A	N/A
2021/0306416	12/2020	Mukhopadhyay et al.	N/A	N/A
2021/0314342	12/2020	Oberg	N/A	H04L 63/20
2021/0320794	12/2020	Auh et al.	N/A	N/A
2021/0329019	12/2020	Shua et al.	N/A	N/A
2021/0334386	12/2020	AlGhamdi et al.	N/A	N/A
2021/0357246	12/2020	Kumar et al.	N/A	N/A
2021/0360032	12/2020	Crabtree et al.	N/A	N/A
2021/0368045	12/2020	Verma	N/A	G06F 11/3664
2021/0382995	12/2020	Massiglia et al.	N/A	N/A
2021/0382997	12/2020	Yi et al.	N/A	N/A
2021/0406365	12/2020	Neil et al.	N/A	N/A
2021/0409486	12/2020	Martinez	N/A	H04L 67/1001
2022/0004410	12/2021	Chen	N/A	N/A
2022/0012771	12/2021	Gustafson	N/A	G06Q 30/0248
2022/0030020	12/2021	Huffman	N/A	N/A
2022/0036302	12/2021	Cella et al.	N/A	N/A
2022/0053011	12/2021	Rao	N/A	G06F 21/577
2022/0060497	12/2021	Crabtree et al.	N/A	N/A
2022/0086173	12/2021	Yavo et al.	N/A	N/A
2022/0100869	12/2021	Berger et al.	N/A	N/A
2022/0131888	12/2021	Kanso	N/A	G06F 21/577
2022/0138512	12/2021	Saillet et al.	N/A	N/A
2022/0156396	12/2021	Bednash et al.	N/A	N/A
2022/0164111	12/2021	Yang et al.	N/A	N/A

2022/0179964	12/2021	Qiao	N/A	G06F 18/29
2022/0182403	12/2021	Mistry	N/A	N/A
2022/0188273	12/2021	Koorapati et al.	N/A	N/A
2022/0197926	12/2021	Passey et al.	N/A	N/A
2022/0210053	12/2021	Du	N/A	H04L 45/12
2022/0215101	12/2021	Rioux et al.	N/A	N/A
2022/0232024	12/2021	Kapoor	N/A	G06F 21/57
2022/0232042	12/2021	Crabtree	N/A	G06F 16/951
2022/0247791	12/2021	Duminuco et al.	N/A	N/A
2022/0263656	12/2021	Moore	N/A	N/A
2022/0284362	12/2021	Bellinger et al.	N/A	N/A
2022/0309166	12/2021	Shenoy et al.	N/A	N/A
2022/0326861	12/2021	Shachar et al.	N/A	N/A
2022/0326941	12/2021	Nelson et al.	N/A	N/A
2022/0327119	12/2021	Gasper et al.	N/A	N/A
2022/0342690	12/2021	Shua	N/A	G06F 11/301
2022/0342997	12/2021	Watanabe et al.	N/A	N/A
2022/0345480	12/2021	Shua	N/A	N/A
2022/0345481	12/2021	Shua	N/A	H04L 67/101
2022/0350931	12/2021	Shua	N/A	H04L 9/0894
2022/0357992	12/2021	Karpovsky	N/A	G06F 9/52
2022/0358233	12/2021	Thakur et al.	N/A	N/A
2022/0360958	12/2021	Cui et al.	N/A	N/A
2022/0374519	12/2021	Botelho et al.	N/A	N/A
2022/0400128	12/2021	Kfir et al.	N/A	N/A
2022/0407841	12/2021	Karpowicz	N/A	H04L 63/0263
2022/0407889	12/2021	Narigapalli et al.	N/A	N/A
2022/0413879	12/2021	Passey et al.	N/A	N/A
2022/0414103	12/2021	Upadhyay et al.	N/A	N/A
2022/0417011	12/2021	Shua	N/A	G06F 21/577
2022/0417219	12/2021	Sheriff	N/A	H04L 67/56
2023/0007014	12/2022	Narayan	N/A	G06F 21/57
2023/0011957	12/2022	Panse et al.	N/A	N/A
2023/0036145	12/2022	Ramachandran et al.	N/A	N/A
2023/0040635	12/2022	Narayan	N/A	G06F 16/90335
2023/0075355	12/2022	Twigg	N/A	H04L 67/306
2023/0087093	12/2022	Ithal et al.	N/A	N/A
2023/0093527	12/2022	Shua	N/A	N/A
2023/0095756	12/2022	Wilkinson	726/6	H04L 63/1416
2023/0110080	12/2022	Hen	726/4	H04L 63/105
2023/0123477	12/2022	Luttwak	726/25	H04L 63/20
2023/0125134	12/2022	Raleigh et al.	N/A	N/A
2023/0134674	12/2022	Quinn et al.	N/A	N/A
2023/0135240	12/2022	Cody et al.	N/A	N/A
2023/0136839	12/2022	Sundararajan et al.	N/A	N/A
2023/0161614	12/2022	Herzberg et al.	N/A	N/A
2023/0161870	12/2022	Herzberg et al.	N/A	N/A

2023/0164148	12/2022	Narayan	726/13	H04L 63/1416
2023/0164164	12/2022	Herzberg et al.	N/A	N/A
2023/0164182	12/2022	Kothari	726/23	H04L 63/1416
2023/0169165	12/2022	Williams	726/1	G06F 21/554
2023/0171271	12/2022	Williams	726/22	H04L 63/1425
2023/0192418	12/2022	Horowitz et al.	N/A	N/A
2023/0208870	12/2022	Yellapragada et al.	N/A	N/A
2023/0224319	12/2022	Isoyama	726/25	G06F 9/45558
2023/0229764	12/2022	Vohra et al.	N/A	N/A
2023/0231867	12/2022	Rampura Venkatachar	726/25	G06Q 10/0635
2023/0237068	12/2022	Sillifant et al.	N/A	N/A
2023/0254330	12/2022	Singh	726/23	G06F 11/323
2023/0297666	12/2022	Atamli et al.	N/A	N/A
2023/0325814	12/2022	Vijayan et al.	N/A	N/A
2023/0336550	12/2022	Lidgi	N/A	H04L 63/10
2023/0336578	12/2022	Lidgi et al.	N/A	N/A
2023/0376586	12/2022	Shemesh et al.	N/A	N/A
2024/0007492	12/2023	Shen et al.	N/A	N/A
2024/0037229	12/2023	Pab?n et al.	N/A	N/A
2024/0045838	12/2023	Reiss et al.	N/A	N/A
2024/0073115	12/2023	Chakraborty	N/A	H04L 41/0654
2024/0080329	12/2023	Reed et al.	N/A	N/A
2024/0080332	12/2023	Ganesh et al.	N/A	N/A
2024/0146818	12/2023	Cody et al.	N/A	N/A
2024/0202359	12/2023	Shukla et al.	N/A	N/A
2024/0241752	12/2023	Crabtree et al.	N/A	N/A
2024/0259396	12/2023	Kerkar et al.	N/A	N/A
2024/0370880	12/2023	Jeske et al.	N/A	N/A
2025/0055870	12/2024	Viswambharan et al.	N/A	N/A
2025/0086280	12/2024	Murphy et al.	N/A	N/A

FOREIGN PATENT DOCUMENTS

Patent No.	**Application Date**	**Country**	**CPC**
106462439	12/2016	CN	N/A
109240804	12/2018	CN	N/A
112989379	12/2020	CN	N/A
2869535	12/2014	EP	N/A
3013016	12/2015	EP	N/A
4160983	12/2022	EP	N/A
4254869	12/2022	EP	N/A
2017120492	12/2016	JP	N/A
2421792	12/2010	RU	N/A
10202009702X	12/2020	SG	N/A

11202103226	12/2020	SG	N/A
2004034184	12/2003	WO	N/A

OTHER PUBLICATIONS

International Search Report of PCT/IB2023/058074, dated Nov. 20, 2023. Searching Authority United States Patent and Trademark Office, Alexandria, Virginia. cited by applicant
Written Opinion of the Searching Authority of PCT/IB2023/058074, dated Nov. 20, 2023. Searching Authority United States Patent and Trademark Office, Alexandria, Virginia. cited by applicant
Ali Gholami; Security and Privacy of Sensitive Data in Cloud Computing: A Survey of Recent Developments; ARIX:2016; pp. 131-150. cited by applicant
Christos Kyrkou; Towards artificial-intelligence-based cybersecurity for robustifying automated driving systems against camera sensor attacks; IEEE 2020; pp. 476-481. cited by applicant
Guo, yu et al. Enabling Encrypted Rich Queries in Distributed Key-Value Stores. IEEE Transactions on Parallel and Distributed Systems, vol. 30, Issue: 6. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8567979 (Year: 2019). cited by applicant
Henry Hanping Feng; Anomaly Detection Using Call Stack Information; IEEE: Year:2003; pp. 1-14. cited by applicant
International Search Report for PCT Application No. PCT/IB2022/060940 dated Feb. 1, 2023. The International Bureau of WIPO. cited by applicant
International Search Report for PCT/IB2023/050848, dated May 9, 2023. International Bureau of WIPO. cited by applicant
International Search Report, PCT/IB23/55312. ISA/US, Commissioner for Patents, Alexandria, Virginia. Dated Aug. 30, 2023. cited by applicant
Kumar, Anuj et al. A New Approach for Security in Cloud Data Storage for IOT Applications Using Hybrid Cryptography Technique. 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control. https://ieeexplore. ieee.org/stamp/stamp.jsp?tp=&arnumber=9087010 (Year: 2020). cited by applicant
Microsoft Build. “Introduction to Azure managed disks”. Aug. 21, 2023, https://docs.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview. cited by applicant
Microsoft Docs. “Create a VM from a managed image”. Article. Jan. 5, 2022. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/create-vm-generalized-managed. cited by applicant
Mishra, Bharati; Jena, Debasish et al. Securing Files in the Cloud. 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7819669 (Year: 2016). cited by applicant
Sahil Suneja; Safe Inspection of Live Virtual Machines; IEEE; Year:2017; pp. 97-111. cited by applicant
Shuvo, Arfatul Mowla et al. Storage Efficient Data Security Model for Distributed Cloud Storage. 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9356962 (Year: 2020). cited by applicant
Written Opinion of the International Searching Authority for PCT Application No. PCT/IB2022/060940 dated Feb. 1, 2023. The International Bureau of WIPO. cited by applicant
Written Opinion of the International Searching Authority, PCT/IB23/55312. ISA/US Commissioner for Patents, Alexandria, Virginia. Dated Aug. 30, 2023. cited by applicant
Written Opinion of the Searching Authority for PCT/IB2023/050848, dated May 9, 2023.

International Bureau of WIPO. cited by applicant
Zhang et al. BMC Bioinformatics 2014. “On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types”. http://www.biomedcentral.com/1471-2105/15/110. cited by applicant
Jordan, M. et al. Enabling pervasive encryption through IBM Z stack innovations. IBM Journal of Research and Development, vol. 62 Issue: 2/3, https://ieeexplore.IEEEieee.org/stamp/stamp.jsp?tp&arnumber=8270590 (Year: 2018). cited by applicant
Leibenger, Dominik et al. EncFS goes multi-user: Adding access control to an encrypted file system. 2016 IEEE Conference on Communications and Network Security (CNS). https://ieeexoplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7860544 (Year: 2016). cited by applicant
Siqi Ma; Certified Copy? Understanding Security Risks of Wi-Fi Hotspot based Android Data Clone Services; ACM; Year: 2021; pp. 320-331. cited by applicant
Chang, Bing et al. MobiCeal: Towards Secure and Practical Plausibly Deniable Encryption on Mobile Devices. 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp= &arnumber=8416506 (Year: 2018). cited by applicant
Islam, Md Shihabul et al. Secure Real-Time Heterogeneous IoT Data Management System. 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). https://ieeexplore.ieee.org/stamp/ stamp.jsp?tp=&arnumber=9014355 (Year: 2019). cited by applicant
Safaryan, Olga A et al. Cryptographic Algorithm Implementation for Data Encryption in DBMS MS SQL Server. 2020 IEEE East-West Design & Test Symposium (EWDTS). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9224775 (Year: 2020). cited by applicant
Wassermann, Sarah et al. ViCrypt to the Rescue: Real-Time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic. IEEE Transactions on Network and Service Management, vol. 17, Issue: 4. https://ieeexplore.ieee.org/stamp/ stamp.jsp?tp=&arnumber=9250645 (Year: 2020). cited by applicant
A Cloud Computing Based Network Monitoring and Threat Detection for Critical Infrastructures, Zhijiang Chen et al., Big Data Research, 2015. cited by applicant
Above the Clouds: A Berkeley View of Cloud Computing, Michael Armbrust et al, Feb. 10, 2009. cited by applicant
Analyzing the Internet's BGP Routing Table, Geoff Huston, Jan. 2001. cited by applicant
AWS, AWS managed policies for job functions, Oct. 26m 2021, https://web.archive.org/web/20211026212847/https:// docs.aws.amazon.com/IAM/latesUUserGuide/access_policiesjob-functions.html (Year: 2021). cited by applicant
Christie Koehler, Detecting and Managing Drift with Terraform, Jun. 7, 2018, https://hasicorp.com/en/blog/detecting-and-managing-drift-with-terraform (Year: 2018). cited by applicant
Cisco Routing Concepts (Jan. 1, 2018). cited by applicant
Cloud Computing Explained, Vangie Beal, May 21, 2010. cited by applicant
Cloud Computing: An Overview, Ling Qian et al., 2009, LNCS 5931, 626-631. cited by applicant
GitHub, Complete EC2 Instance, Aug. 27, 2021, https://github.com/terraform-aws-modules/terraform-aws-ec2-instance/tree/ 528613d4580f2c1266e87d8d24fc25bf5290fe2c/examples/complete (Year: 2021). cited by applicant
GitHub, Complete EC2 Instance, Aug. 27, 2021, https://github.com/terraform-aws-modules/terraform-aws-ec2-instance/tree/

528613d4580f2c1266e87d8d24fc25bf5290fe2c/examples/complete/main.tf (Year: 2021). cited by applicant
How Does the Internet Work? Rus Shuler, Pomeroy IT Solutions, 2002. cited by applicant
Interney Protocol, DARPA Internet Program Protocol Specification, Sep. 1981, RFC: 791. cited by applicant
Internet Protocol, Version 6 Specification, Dec. 1995. cited by applicant
Introduction to Cloud Computing, Cloud Computing I, 15-319, Spring 2010, Majd F. Sakr. cited by applicant
Network Attack Surface: Lifting the Concept of Attack Surface to the Network Level for Evaluating Networks'Resilience Against Zero-Day Attacks, Mengyuan Zhang et al., IEEE vol. 18, No. 1, Jan./Feb. 2021. cited by applicant
No Vacancy: IPv4 Address Depletion and Possible Solutions for the Expanding Internet, Illumin Magazine, Jun. 27, 2011, Steve Wolfsohn. cited by applicant
Orca Security Ltd, v. *Wiz, Inc*., Inter Partes Review Case No. IPR2025-01085; U.S. Pat. No. 11,936,693. cited by applicant
PCI Data Security Standard (PCI DSS), Version 1.1, Sep. 2017. cited by applicant
PCI Data Security Standard (PCI DSS), Version 1.2, Mar. 2008. cited by applicant
PCI Data Security Standard Requirements and Testing Procedures, Version 4.0, Mar. 2022. cited by applicant
Proxify, Mastering good programming practices: A comprehensive guide, Apr. 27, 2021, https://proxify.io/articles/good-programming-practices (Year: 2021). cited by applicant
Reddit, 1AM Roles for each Lambda?, Aug. 25, 2019 (Year: 2019). cited by applicant
Study: Cloud Computing to Brighten Future of Data Centers, Martin LaMonica, Mar. 10, 2008, CNET. cited by applicant
Transmission Control Protocol, DARPA Internet Program Protocol Specification, Sep. 1981, RFC: 793. cited by applicant
Understanding Cloud Computing Basics, Oct. 9, 2020, EU Online. cited by applicant
Use of Cloud Computing Applications and Services, John Horrigan, Sep. 12, 2008. cited by applicant

---

Primary Examiner: Zhao; Don G

Attorney, Agent or Firm: M&B IP Analysts, LLC

Background/Summary

TECHNICAL FIELD
(1) The present disclosure relates generally to exposure detection in cloud environments, and specifically to active detection of exposure in cloud environments.
BACKGROUND
(2) External attack surface management (EASM) is a term which for a technology field and best practices which are utilized in cybersecurity to describe what vulnerabilities an organization has within their network infrastructure, which may include cloud computing environments, local network environments, and the like. For example, an organization may have a virtual private cloud (VPC) implemented in Amazon® Web Services (AWS), Microsoft® Azure, Google® Cloud Platform (GCP), and the like, which serves as a cloud computing environment. The cloud computing environment may include a plurality of workloads, such as virtual machines, container engines, serverless functions, and the like, any of which may pose a security risk, for example by

having a vulnerability, allowing an attacker to infiltrate the organization's network in an unintended manner.
(3) EASM technologies aim to discover where an organization is vulnerable, in order for a network administrator to secure the discovered vulnerabilities. For example, discovering an out-of-date operating system (OS) having a known vulnerability running on a virtual machine may require the network administrator to update the OS version, or apply a software patch, in order to address the vulnerability. This is also known as minimizing the external attack surface.
(4) One such technology which may be deployed in order to discover the external attack surface is known is active scanning. Active scanning attempts to infiltrate a network (e.g., access resources in the above mentioned VPC). For example, by sending packets to endpoints in the network. Thus, an active scanner may attempt to access random domains, at random ports, in order to gain access to a network or to a network resource.
(5) This method has some serious drawbacks. For example, attempting to guess random domains, random ports, and the like, creates a large volume of network traffic which the target (i.e., organization's network) must deal with. This may congest the network, and further risks malfunctions, such as a denial of service to other clients, data corruption from incompatible queries, and the like. It is often of upmost importance to an organization to keep a production environment in a fully operational state. Therefore, using an active scanner to test accessibility of an active production environment may be detrimental to this objective, since it would require devotion of substantial resources at least in terms of network bandwidth to perform such tests.
(6) It would therefore be advantageous to provide a solution that would overcome the challenges noted above.
SUMMARY
(7) A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term “some embodiments” or “certain embodiments” may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.
(8) Certain embodiments disclosed herein include a method for performing active inspection of a cloud computing environment. The method comprises: receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from a network which is external to the cloud computing environment; and actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from a network external to the cloud computing environment.
(9) Certain embodiments disclosed herein also include a non-transitory computer readable medium having stored thereon causing a processing circuitry to execute a process, the process comprising: receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from a network which is external to the cloud computing environment; and actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from a network external to the cloud computing environment.
(10) Certain embodiments disclosed herein also include a system for performing active inspection of a cloud computing environment. The system comprises: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: receive at least one network path to access a first resource, wherein the first resource is a

cloud object deployed in the cloud computing environment, and potentially accessible from a network which is external to the cloud computing environment; and actively inspect the at least one network path to determine if the first resource is accessible through the at least one network path from a network external to the cloud computing environment.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
(1) The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.
(2) FIG. **1** is a diagram of a cloud computing environment monitored by an active inspector, implemented in accordance with an embodiment.
(3) FIG. **2** is a security graph illustrating a network path, implemented in accordance with an embodiment.
(4) FIG. **3** is a flowchart of a method for performing active inspection of a cloud computing environment, implemented in accordance with an embodiment.
(5) FIG. **4**A is a flowchart depicting a method for determining reachable properties of security objects, according to an embodiment.
(6) FIG. **4**B is a flowchart depicting the analysis of a network path to determine reachable properties of objects included in the path, according to an embodiment.
(7) FIG. **5** is a screenshot generated by an active inspector, implemented in accordance with an embodiment.
(8) FIG. **6** is a schematic diagram of an active inspector **125** according to an embodiment.
DETAILED DESCRIPTION
(9) It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.
(10) The various disclosed embodiments include A system and method for performing active inspection of a cloud computing environment includes receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from a network which is external to the cloud computing environment; and actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from a network external to the cloud computing environment.
(11) Various techniques of static analysis can be used in order to determine reachability properties of a resource deployed in a cloud computing environment. Reachability properties, or parameters, may be utilized to establish a network path to the resource from an external network through the cloud computing environment. An access instruction may be generated based on the network path to determine if a network path generated through static analysis is indeed a viable path to reach the resource. Determining what network paths are viable is advantageous as it exposes what network paths can be used to access the cloud computing environment from external networks, and therefore what parts of the cloud computing environment are in practice opened to attack. These network paths should be addressed by system administrators as early as possible to minimize the effect of a cyber-attack.

(12) FIG. **1** is an example diagram **100** of a cloud computing environment monitored by an active inspector, implemented in accordance with an embodiment. A first cloud environment **110** includes a plurality of principals and resources. A resource is a cloud entity which supplies functionality, such as processing power, memory, storage, communication, and the like. A resource may supply more than one functionality. Resources may include, for example, virtual machines (VMs) such as VMs **113**, container engines such as container engines **115**, serverless functions such as serverless functions **117**, and the like. A VM may be implemented using Oracle® VirtualBox. A container engine may be implemented using Kubernetes® or Docker®. A serverless function may implemented using Lambda®.
(13) A principal is a cloud entity which acts on a resource, meaning it can request, or otherwise initiate, actions or operations in the cloud environment which cause a resource to perform a function. A principal may be, for example, a user account such as user account **112**, a service account such as service account **114**, a role, and the like. In an embodiment a user account **112** is implemented as a data structure which includes information about an entity, such as username, a password hash, an associated role, and the like.
(14) The first cloud environment **110** may be implemented utilizing a cloud infrastructure, such as Amazon® Web Services (AWS), Microsoft® Azure, Google® Cloud Platform (GCP), and the like. In an embodiment, the first cloud environment **110** may be implemented as a virtual private cloud (VPC) on such a cloud infrastructure. The first cloud environment **110** may be, for example, a production environment for an organization. A production environment is a computing environment which provides services, for example, to client devices within the production environment and outside of it. An organization may also have a staging environment, which is a computing environment substantially identical to the production environment in at least some deployments of resource (e.g., workloads) which is used for the purpose of testing new policies, new permissions, new applications, new appliances, new resources, and the like, which are not present in the production environment.
(15) It is often of upmost importance to an organization to keep the production environment in a fully operational state. Therefore, using an active scanner to test accessibility to the first cloud environment **110** may be detrimental to this objective, since it would require devotion of substantial resources at least in terms of network bandwidth to perform such tests.
(16) An inspection environment **120** is communicatively connected with the first cloud environment **110**, and a public network **130**. The public network **130** is also communicatively connected with the first cloud environment **110**. In an embodiment, the public network **120** may be, but is not limited to, a wireless, cellular or wired network, a local area network (LAN), a wide area network (WAN), a metro area network (MAN), the Internet, the worldwide web (WWW), similar networks, and any combination thereof.
(17) The inspection environment **120** may be implemented as a VPC in a cloud infrastructure. In an embodiment, the cloud infrastructure of the inspection environment **120** may be the same cloud infrastructure as the first cloud environment **110**. In some embodiments, the inspection environment may be implemented as multiple cloud environments, each utilizing a cloud infrastructure. The inspection environment includes a security graph database (DB) **122** for storing a security graph, and at least an active inspector **125**.
(18) In an embodiment, the security graph stored in the security graph DB **122** represents at least the first cloud environment **110** using a predefined data schema. For example, each resource and each principal of the first cloud environment **110** may be represented as a corresponding resource node or principal node in the security graph. The various nodes in the security graph may be connected, for example, based on policies, roles, permissions, and the like, which are detected in the first cloud environment **110**. A predefined data schema may include data structures including into which values can be inputted to represent a specific cloud entity. For example, a resource may be represented by a template data structure which includes data attributes, whose values uniquely

identify the resource, such as address, name, type, OS version, and the like.
(19) The active inspector **125** is configured to receive a network path to access a resource in the first cloud environment **110**. In an embodiment, a network path may be stored as a data string which includes one or more reachability parameters. Such parameters include host names, protocols, IP addresses, ports, usernames, passwords, and the like. In certain embodiments, the active inspector **125** is further configured to receive a list of network paths. The network paths may be received periodically. In certain embodiments, the active inspector **125** is also configured to generate an instruction which includes a query for the security graph, such instruction or instructions when executed by the security graph database **122** cause(s) generation of an output including one or more network paths. For example, network paths may be generated every 24 hours, while active inspection may occur once per day, once per week, once per month, and so on.
(20) An example of a static analysis process for generating network paths, also known as determining reachability to a resource, is discussed in more detail in U.S. Non-Provisional patent application Ser. No. 17/179,135 filed on Feb. 18, 2021, the contents of which are hereby incorporated by reference herein. In an embodiment, the active inspector **125** may generate an instruction based on the network path to access the resource associated with the network path. For example, the instruction may be to send a data packet to an IP address of the resource, and receive an acknowledgement (ACK) response. The active inspector **125** may generate a log which includes, for example, the network path, the instruction sent by the active inspector **125**, and any response(s) received from the resource. For example, if the active inspector **125** sends an HTTP (hypertext transfer protocol) request, a response may be a 404 error, a 403 error, 500 error, 502 error, and the like.
(21) In an embodiment the active inspector **125** initiates active inspection of a network path to determine if a resource is accessible via the network path from a network which is external to the first cloud environment **110**.
(22) FIG. **2** is an example of a security graph **200** illustrating a network path, implemented in accordance with an embodiment. The security graph **200** includes a plurality of nodes, each node connected to at least another node by an edge. In certain embodiments, a pair of nodes may be connected by a plurality of edges. In some embodiments, each edge may indicate a type of connection between the nodes. For example, an edge may indicate a “can access”, to indicate that a cloud entity represented by a first node can access the cloud entity represented by a second node.
(23) A first enrichment node **210** (also referred to as public network node **210**) represents a public network, such as public network **130** of FIG. **1** above. An enrichment node, such as enrichment node **210**, is a node generated based off of insights determined from data collected from a computing environment, such as the first cloud computing environment **110** of FIG. **1** above. An enrichment node may also represent, for example, a vulnerability. By connecting resource nodes in the graph to the enrichment node representing a vulnerability, the security graph **200** may indicate that the resources contain the vulnerability. This allows a compact representation as the security graph does not redundantly store multiple data fields of the same vulnerability in each resource node.
(24) The public network node **210** is connected to a first resource node **220** (also referred to as firewall node **220**) representing a firewall workload. The firewall represented by the firewall node **220** may be implemented, for example, as a virtual machine in the first cloud computing environment. Connecting the public network node **210** to the firewall node **220** represents that the firewall is open to transceiving communication between itself and the public network.
(25) The firewall node **220** is further connected to a second resource node **230** (also referred to as API gateway node **230**) which represents an API (application programming interface) gateway. An API gateway is a workload, for example a serverless function, which can act as a reverse proxy between a client and resources, accepting API calls, directing them to the appropriate service, workload, resource, etc. and returning a result to the client when appropriate.

(26) The API gateway node **230** is connected to a first principal node **240** (also referred to as VM node **240**) representing a virtual machine hosting an application and a database, and is also connected to a second principal node **250** (also referred to as container engine node **250**) which hosts a plurality of container nodes. The VM node **240** is connected to an application node **242**, and a database node **244**. The application node **242** may indicate, for example, that a certain application, having a version number, binaries, files, libraries, and the like, is executed on the VM which is represented by the VM node **240**.
(27) In an embodiment, the VM node **240** may be connected to a plurality of application nodes. The database node **244** represents a database which is stored on the VM (represented by VM node **240**), or stored on a storage accessible by the VM. The database node **244** may include attributes which define a database, such as type (graph, columnar, distributed, etc.), version number, query language, access policy, and the like.
(28) FIG. **3** is an example flowchart **300** of a method for performing active inspection of a cloud computing environment, implemented in accordance with an embodiment.
(29) At S**310**, at least one network path for a first resource in a cloud computing environment is received. The network path, also known as object reachability, includes data (e.g. reachability parameters) for accessing the first resource from a public network, which is not the cloud computing environment of the first resource, such as the Internet. In an embodiment, an active inspector may receive the at least a network path, for example from a security graph. In an embodiment, S**320** includes generating an instruction (or instructions) which when executed by a database system storing the security graph return a result of one or more resources, and a respective network path for each of the one or more resources. In certain embodiments, the network paths may be received periodically.
(30) In some embodiments, the first resource may be one of a plurality of first resources, which are each substantially identical. For example, a group of virtual machines which are generated based on the same code or image are substantially identical, since their initial deployment would be identical other than a unique identifier assigned to each machine. In such embodiments it may be beneficial to inspect the at least one network path for a subset of the plurality of first resources, in order to decrease the computation and network resources required. This may be acceptable in such embodiments, as the expectation is that the plurality of VMs would be accessible in similar network paths. In some embodiments, the subset includes one or more first resources.
(31) In an embodiment, each of the received network paths includes a set of reachability parameters to reach a specific cloud object in the cloud environment. The reachability parameters, and hence the network paths are generated by statically analyzing the cloud environment. An example method for such static analysis is described with reference to FIGS. **4**A and **4**B below.
(32) At S**320**, an access instruction is generated to access the first resource based on the network path. In an embodiment, the access instruction is generated by the active inspector deployed outside of the cloud environment where the first resource resides. In certain embodiments, the instruction includes one or more access parameters. Such parameters may include, but are not limited to, a host name, an IP address, a communication protocol, a port, a username, a password, and the like, or combination thereof. A communication protocol may be, for example, HTTP or UDP (user datagram protocol). For example, the instruction may be a ping, GET, CONNECT, or TRACE request over HTTP.
(33) In certain embodiments, a plurality of access instructions may be generated. For example, a plurality of generated access instructions may include a first access instruction having a first request, and a second access instruction having a second request which is different from the first request. For example, the first access instruction may include a CONNECT request, and the second access instruction may include a GET request. In certain embodiments, a plurality of first access instructions may be generated. In such embodiments, each first access instruction may include a same type of request (e.g., CONNECT) with different values (e.g., different web address, different

port, and so on). For example, a resource may be reachable at IP address 10.0.0.127, at ports **800** through **805**. The IP address and ports would be reachability parameters, based on which an active inspector can generate a plurality of first access instructions based on an HTTP GET request, such as: GET/bin HTTP/1.1 Host: 10.0.0.127:800
and further generate another HTTP GET request: GET/bin HTTP/1.1 Host: 10.0.0.127:801
and so on, which when executed attempt to access a/bin folder in the resource which has an IP address of 10.0.0.127. In certain embodiments, the active inspector (e.g., the active inspector **125** of FIG. **1**) may connect to a proxy server (not shown) through the public network **130**, and send a first access instruction to a resource in the cloud environment **110** through a first proxy server, and send a second access instruction (which may or may not be identical to the first access instruction) through a second proxy server. In such embodiments, each proxy server may show as originating from a different country of origin, therefore the source would receive access requests from seemingly different sources. This is advantageous to determine, for example, if a resource is configured to block certain network traffic based on geographic location.
(34) At S**330**, execution of the generated access instruction is caused. The access instruction, when executed, causes an attempt to actually access the resource. In an embodiment, the attempt may result in network traffic being generated, including requests sent to the resource and answers (i.e., data packets) received. While static analysis provides a possible path to access a resource, executing the access instruction provides a real result of an attempt to utilize the possible path, in order to determine which paths are really viable, and which are not. For example, a path may be possible based on static analysis, but not viable, where, for example, an application deployed on the resource prevents such an access from occurring. In an embodiment a network path is determined to be viable (or accessible), if the access instruction, when executed does not return an error message. An error message may be, for example, a timeout (e.g., in response to a “ping” request), a **403** Forbidden (e.g., in response to an HTTP GET request), and the like. In some embodiments, the access instruction may be executed by the active inspector **125**.
(35) At S**340**, a determination is performed to determine if the network path is accessible, based on the execution of the generated access instruction. Performing an active inspection of a cloud environment allows to determine which of the reachability paths (i.e., network paths) are indeed vulnerable, meaning that paths that can be used to gain access into the cloud environment, and which reachability paths (network paths) are not vulnerabilities since the active inspector could not gain access to the resource, therefore the reachability path is not possible in practice. Reachability paths which have been confirmed through both static analysis (i.e., analysis using the security graph) and active inspection are paths which should therefore be considered more vulnerable. In an embodiment, if the network path results in successfully reaching the resource, the network path is determined to be accessible (or viable). If the resource is not reachable by the network path, the network path is determined to be inaccessible (or unviable).
(36) At S**350**, a security graph is updated based on the network path determination. In certain embodiments, the active inspector may update the security graph, which includes a representation of the cloud environment in which the first resource is deployed, to indicate whether a reachability path is confirmed (i.e., is viable) by active inspection or not, where a confirmed path is a path through which the active inspector successfully accessed a resource. In turn, the security graph may update an alert generated based on determining that a resource has a reachability path through a public network.
(37) At S**360**, a report is generated based on the execution of the generated instruction. In an embodiment, the report may be generated by the active inspector, which performs this method. In certain embodiments, generating a report may include updating a log with network traffic between the active inspector and the resource. For example, the active inspector may record (e.g., write to a log) the generated instruction, the resource identifier, and a response received from the resource. A response may include, for example, a response code. A response code may indicate success,

redirection, client error, server error, and the like, where the client is the active inspector, and the server is the resource. In certain embodiments the security graph stored in the security DB **122** may be updated based on the determined viability of the network paths. For example, if a resource is successfully accessed, or successfully unaccessed (i.e., an attempt was made to access the resource and the attempt was not successful in accessing the resource), this result can be stored as an attribute of a node representing the resource in the security graph. For example, the VM node **240** of FIG. **2** may have an attribute which indicates a reachability status, which may have values corresponding to: successfully reached (i.e., an active inspector successfully accessed this resource), successfully not reach (i.e., an active inspector was not successful in accessing this resource), and undetermined (the active inspector has not yet attempted to access the resource through a network path). In some embodiments, certain network paths may be determined (i.e., as viable or unviable) while others may be undetermined. A node may be associated with a plurality of network paths, each having its own active inspection indicator.
(38) In some embodiments, the active inspector may communicate with a virtual private network (VPN) or a proxy, in order to mask the IP address from which the active inspector is attempting access. This may be useful to test, for example, if a firewall, such as represented by the firewall node **220** of FIG. **2**, will let communication through based on blocking or allowing certain IP addresses. In such embodiments, multiple similar instructions may be generated, each originating from a different IP address of the active inspector.
(39) In some embodiments network path may include a plurality of resources. The method above may be performed on each resource of the plurality of resources, to determine the reachability of each resource.
(40) Utilizing an active inspector using network paths generated from a security graph is advantageous, as attempting to access resources in this manner to determine the viability of a network path (i.e., reachability) requires less resources than, for example, randomly guessing network paths in an attempt to access resources.
(41) In certain embodiments the active inspector may generate a screenshot of a user interface used to access the resource through the network path. FIG. **5** below is one such example of a screenshot of a user interface, implemented in accordance with an embodiment.
(42) Furthermore, utilizing the active inspector to validate network paths and updating the security graph with the results allows to detect workloads which both contain a vulnerability, and have a validated network path. This allows generating an alert to a user of the cloud environment in order to address such problems by accurately characterizing cybersecurity threats. This in turn allows to utilize resources more efficiently, since the most vulnerable gaps in the cloud environment will be addresses first.
(43) FIG. **4**A is an example flowchart **400** depicting a method for determining reachable properties of security objects, according to an embodiment. A reachable property defines if and how an object on the generated security graph can be reached from an external or internal network, and/or an external or internal object. External means outside of the cloud environment of an organization. An object may be any computing or network object designated in a security graph generated as discussed above.
(44) At S**405**, a security graph is accessed or otherwise obtained from the graph database. Within a security graph, various objects or entities, as may be included in a network or cloud environment of an organization, may be represented as “nodes” or “vertices,” and such “nodes” or “vertices” may be interconnected by one or more “links” or “edges,” the “links” or “edges” representing the relationships between the various objects included in a network or environment. Each object in the graph may be associated with known properties of the object. Examples for such properties may include an object's name, its IP address, various predefined security rules or access rules, and the like.
(45) At S**410**, possible network paths within the obtained security graph are identified. A network

path is a connection of two or more security objects accessible from an external or internal network, and/or an external or internal object. That is, a network path may include sequential representations of possible data/control flows between two or more objects in a graph. In an embodiment, where two objects in a graph are represented as vertices, and where the vertices are joined by an edge, a path may be constructed between the two vertices. A path may be a vertex-only path, describing a sequence of vertex-to-vertex "hops," an edge-only path, describing only the edges included in the sequence without description of the associated vertices, or a combined edge-vertex path, describing both edges and vertexes included in the sequence.
(46) According to disclosed embodiments, a path shows a connection between security objects and/or computing objects that communicate over a network. An object may be a virtual, physical, or logical entity.
(47) In an embodiment, paths can be identified by traversing the security graph. The traversal can start or end at objects that are connected to an external network (the internet). The traversal of the security graph can be performed using solutions disclosed in the related art, e.g., a breadth-first search (BFS), a tree traversal, and the like, as well as any combination thereof.
(48) In another embodiment, paths can be identified by querying the graph database storing the security graph. Examples of applicable queries include, without limitation, queries configured to identify all paths between a first graph object (node) and a second graph object, queries configured to identify all paths between all graph vertices of a first object type and all graph vertices of a second object type, other, like, queries, and any combination thereof.
(49) Following as performed at S**410** through S**430**, the list of paths are iteratively identified to determine the reachability properties of the path. Specifically, at S**415**, a path list is populated to include all identified paths. A path list may be a table, list, or other type of data structure. A path list may be unordered or ordered, including ordering according to one or more path properties.
(50) At S**420**, a path from the path list is selected. At a first run of the method a first path in the list is selected.
(51) At S**425**, path elements are analyzed to determine reachable properties. Path element analysis, as at S**425**, is an iterative analysis of each element included in the path selected at S**420**. The operation of S**425** is discussed in detail with reference to FIG. **4**B.
(52) At S**430**, it is determined whether the last path of the path list has been analyzed, and if so, execution terminates; otherwise, execution returns to S**420**.
(53) FIG. **4**B is an example flowchart S**425** depicting the analysis of a network path to determine reachable properties of objects included in the path, according to an embodiment.
(54) At S**455**, elements within a selected network path are identified. Elements are network and/or computing objects and relationships (or connections) between such objects. Identification of elements within the selected path may include, without limitation, identification based on properties, and other, like, data, included in the elements, identification of elements based on element identifications provided during the execution of S**410** of FIG. **4**A, above, and the like, as well as any combination thereof. Further, identification of in-path elements may include identification of element properties or attributes including, without limitation, names, network addresses, rulesets, port configurations, and the like, as well as any combination thereof.
(55) Then, at S**460** through S**480**, the list of paths are iteratively processed in order to determine reachable properties of the elements. Specifically, at S**460**, the next element is selected. The next element is a subsequent element of the set of elements, within the selected path, identified at S**455**. Where execution of S**460** follows the execution of S**480**, the next element may be an element which, in the selected network path, immediately follows the element relevant to the preceding execution of S**470** and S**475**. Where execution of the method described with respect to FIG. **4**B includes a first execution of S**460**, the first execution of S**460** may include the selection of a first element of the selected path.
(56) For exemplary purposes, a network path may be a path from a virtual machine (VM),

connected to a NIC, connected to a load balancer, connected to a firewall. According to a first example, where S**460** is executed for the first time, the first execution of S**460** may include the selection of the VM as the selected element. Further, according to a second example, where execution of S**460** follows execution of S**480**, selection of a next element at S**460** may include selection of, following the VM, selection of the NIC, or, following the NIC, selection of the load balancer, or, following the load balancer, selection of the firewall.
(57) At S**465**, it is determined whether the selected element has been analyzed. Determination of whether the selected element may include the determination of whether one or more reachable properties are included in the relevant graph element. As execution of S**475** provides for the population of reachable properties into the security graph, an element which does not include such reachable properties in the graph may be assumed to have not been analyzed.
(58) Where, at S**465**, it is determined that the selected element has been analyzed, execution continues with S**460**. Where, at S**465**, it is determined that the selected element has not been analyzed, execution continues with S**470**.
(59) At S**470**, reachable properties are determined. Reachable properties are object properties describing if, and how, a given path element is reachable through the selected path, and, specifically, from an external network, an internal network, both, and a combination thereof. Examples of reachable properties include, without limitation, binary properties describing whether an element is reachable, protocols by which the element is reachable, network addresses at which an element is reachable, ports by which an element is reachable, access rules, and the like, as well as any combination thereof.
(60) In an embodiment, a reachable property is determined as a minimal set of reachable properties of all other objects in the path. As a simple example, if a path includes two objects, where one object can receive traffic from any source IP address through port **1515**, and the other object can receive traffic only from a source IP address of 173.54.189.188, the reachable property of the second object may be that the second object is reachable through “source IP address 173.54.189.188 and port **1515**.”
(61) At S**475**, reachable properties are populated into the security graph. Reachable properties, as may be determined at S**470**, may be populated into the graph by processes including, without limitation, labeling or tagging graph vertices (or “nodes”), updating network or graph object properties, generating one or more graph overviews, layers, or graph-adjacent data features, and the like, as well as any combination thereof.
(62) In an embodiment, population of reachable properties into the security graph may include, for each object, population of object network access control lists (NACLs) as described hereinbelow, into the security graph elements corresponding with the various path elements, as well as the population of scope specific NACLs, and other, like, properties into the graph. Scope-specific NACLs are NACLs describing object, path, or network accessibility properties specific to a given scope, where a given scope may be the internet, various given accounts, various given environments, and the like. Scope-specific NACLs may, for example, describe the properties of an object with respect to the object's internet accessibility, where the object may be configured to include different access control properties for internet access and local intranet access.
(63) Further, population of reachable properties into the graph may include population of one or more paths into the graph, including by population processes similar or identical to those described with respect to population of individual objects. Population of paths into the graph may include, without limitation, population of one or more paths into the graph, including a presently-analyzed path, population of one or more path properties, and the like, as well as any combination thereof. Path properties, as may be populated to a graph, are properties describing various attributes of a path, including, without limitation, NACLs applicable to path elements, path segments, or full paths, including full-path aggregate NACLs, and the like, as well as any combination thereof. Further, population of path properties into the graph may include the population of one or more

scope-specific path properties, where such scope-specific path properties may be properties relevant to specific scopes, such as those described herein.
(64) Where population of reachable properties includes labeling or tagging a graph, or elements thereof, one or more graph vertices or edges, the corresponding objects or relationships, or both, may be labeled, tagged, or otherwise associated with one or more data features describing relevant reachable properties. In addition, where population of reachable properties to the graph includes updating graph objects, graph vertices and edges, the corresponding objects and relationships, or both, may be directly updated to explicitly include the calculated properties.
(65) Further, where population of reachable properties includes the generation of one or more graph layers or overlays, the generated graph layers or overlays may be data features independent of, but corresponding to, the relevant graphs, where the generated overlays or layers may include one or more data features describing the reachable properties of the various graph elements.
(66) At S**480**, it is determined whether all elements in the selected path have been analyzed. Determination of whether all elements in the selected path have been analyzed may include, without limitation, determination of whether the immediately preceding execution of S**475** relates to the last element in the selected path, determination of whether additional elements remain in the path, determination of whether any additional in-path elements have been analyzed, and the like, as well as any combination thereof.
(67) Where, at S**480**, it is determined that all elements in the selected path have not been analyzed, execution continues with S**460**. Where, at S**480**, it is determined that all elements in the selected path have been analyzed, execution terminates.
(68) FIG. **5** is an example of a screenshot **500** generated by an active inspector, implemented in accordance with an embodiment. A screenshot is an image which shows the contents of a computer display. In an embodiment, an active inspector, such as the active inspector **125** of FIG. **1**, may include a web browser application for executing access instructions. The web browser application may generated a user interface intended for a display. The screenshot **500** includes a portion of such a user interface, which includes a response header **510** received based on a request to access a resource. In this case the response header **510** includes an HTTP code **403** (i.e., forbidden), meaning that the request to access the resource was denied. A detailed code **512** includes a message which is associated with the **403** code (i.e., “access denied”), a message **514**, a request identifier **516**, and a host identifier **518**.
(69) FIG. **6** is an example schematic diagram of an active inspector **125** according to an embodiment. The active inspector **125** includes a processing circuitry **610** coupled to a memory **620**, a storage **630**, and a network interface **640**. In an embodiment, the components of the active inspector **125** may be communicatively connected via a bus **650**.
(70) The processing circuitry **610** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.
(71) The memory **620** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof.
(72) In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **630**. In another configuration, the memory **620** is configured to store such software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by

the processing circuitry **610**, cause the processing circuitry **610** to perform the various processes described herein.
(73) The storage **630** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, compact disk-read only memory (CD-ROM), Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.
(74) The network interface **640** allows the active inspector **125** to communicate with, for example, a cloud environment, a security graph database, resources from the cloud environment, and the like.
(75) It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **6**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.
(76) The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.
(77) All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.
(78) It should be understood that any reference to an element herein using a designation such as "first," "second," and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.
(79) As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; 2A; 2B; 2C; 3A; A and B in combination; B and C in combination; A and C in combination; A, B, and C in combination; 2A and C in combination; A, 3B, and 2C in combination; and the like.

Claims

1. A method for performing active inspection of a cloud computing environment, comprising: receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from an external network which is external to the cloud computing environment; actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from the external network by sending a data packet over the at least one network path to the first resource; actively inspecting if a subset of a plurality of second resources is accessible through another network path from a network external to the cloud computing environment; and determining that each of the second resources is accessible through the another network path from the network external to the cloud computing environment when the executed instruction does not return an error for the subset of the plurality of second resources.
2. The method of claim 1, further comprising: generating a first instruction to access the first resource based on a plurality of reachability parameters designated in the at least one network path; causing execution of the generated first instruction to access the first resource; and determining that the at least a network path is accessible from the external network when the executed instruction does not return an error.
3. The method of claim 2, further comprising: generating a report based on executing the generated instruction, the generated report including network traffic between the first resource and an active inspector.
4. The method of claim 2, further comprising: generating a plurality of first instructions, each first instruction differing from another first instruction by a value of a reachability parameter.
5. The method of claim 2, wherein the first instruction is executed via a first external network associated with a first IP address, and another first instruction is executed via a second external network associated with a second IP address, which is different from the first IP address.
6. The method of claim 2, wherein the first instruction utilizing any one of: HTTP, and UDP.
7. The method of claim 2, wherein the first instruction includes any one of: ping, get, connect, trace, and any combination thereof.
8. The method of claim 2, wherein the reachability parameters are any one of: an IP address, a host name, a user name, a password, a port, a web address, a communication protocol, and any combination thereof.
9. The method of claim 1, further comprising: updating a security graph based on a result of active inspection, wherein the security graph includes a representation of the cloud computing environment.
10. A system for performing active inspection of a cloud computing environment, comprising: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: receive at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from an external network which is external to the cloud computing environment; and actively inspect the at least one network path to determine if the first resource is accessible through the at least one network path from the external network by sending a data packet over the at least one network path to the first resource; actively inspect if a subset of a plurality of second resources is accessible through another network path from a network external to the cloud computing environment; and determine that each of the second resources is accessible through the another network path from the network external to the cloud computing environment when the executed instruction does not return an error for the subset of the plurality of second resources.
11. The system of claim 10, wherein the memory further contains instructions that when executed by the processing circuitry further configures the system to: generate a first instruction to access the first resource based on a plurality of reachability parameters designated in the at least one network path; execute the generated first instruction to access the first resource; and determine that the at

least a network path is accessible from the external network when the executed instruction does not return an error.
12. The system of claim 11, wherein the memory further contains instructions that when executed by the processing circuitry further configures the system to: generate a report based on executing the generated instruction, the generated report including network traffic between the first resource and an active inspector.
13. The system of claim 11, wherein the memory further contains instructions that when executed by the processing circuitry further configures the system to: generate a plurality of first instructions, each first instruction differing from another first instruction by a value of a reachability parameter.
14. The system of claim 11, wherein the first instruction is executed via a first external network associated with a first IP address, and another first instruction is executed via a second external network associated with a second IP address, which is different from the first IP address.
15. The system of claim 11, wherein the first instruction utilizing any one of: HTTP, and UDP.
16. The system of claim 11, wherein the first instruction includes any one of: ping, get, connect, trace, and any combination thereof.
17. The system of claim 11, wherein the reachability parameters are any one of: an IP address, a host name, a user name, a password, a port, a web address, a communication protocol, and any combination thereof.
18. The system of claim 10, wherein the memory further contains instructions that when executed by the processing circuitry further configures the system to: update a security graph based on a result of active inspection, wherein the security graph includes a representation of the cloud computing environment.
19. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process, the process comprising: receiving at least one network path to access a first resource, wherein the first resource is a cloud object deployed in the cloud computing environment, and potentially accessible from an external network which is external to the cloud computing environment; and actively inspecting the at least one network path to determine if the first resource is accessible through the at least one network path from the external network external by sending a data packet over the at least one network path to the first resource; actively inspect if a subset of a plurality of second resources is accessible through another network path from a network external to the cloud computing environment; and determine that each of the second resources is accessible through the another network path from the network external to the cloud computing environment when the executed instruction does not return an error for the subset of the plurality of second resources.