Trickbot rdpscanDll – Transforming Candidate Credentials for Brute-Forcing RDP Servers

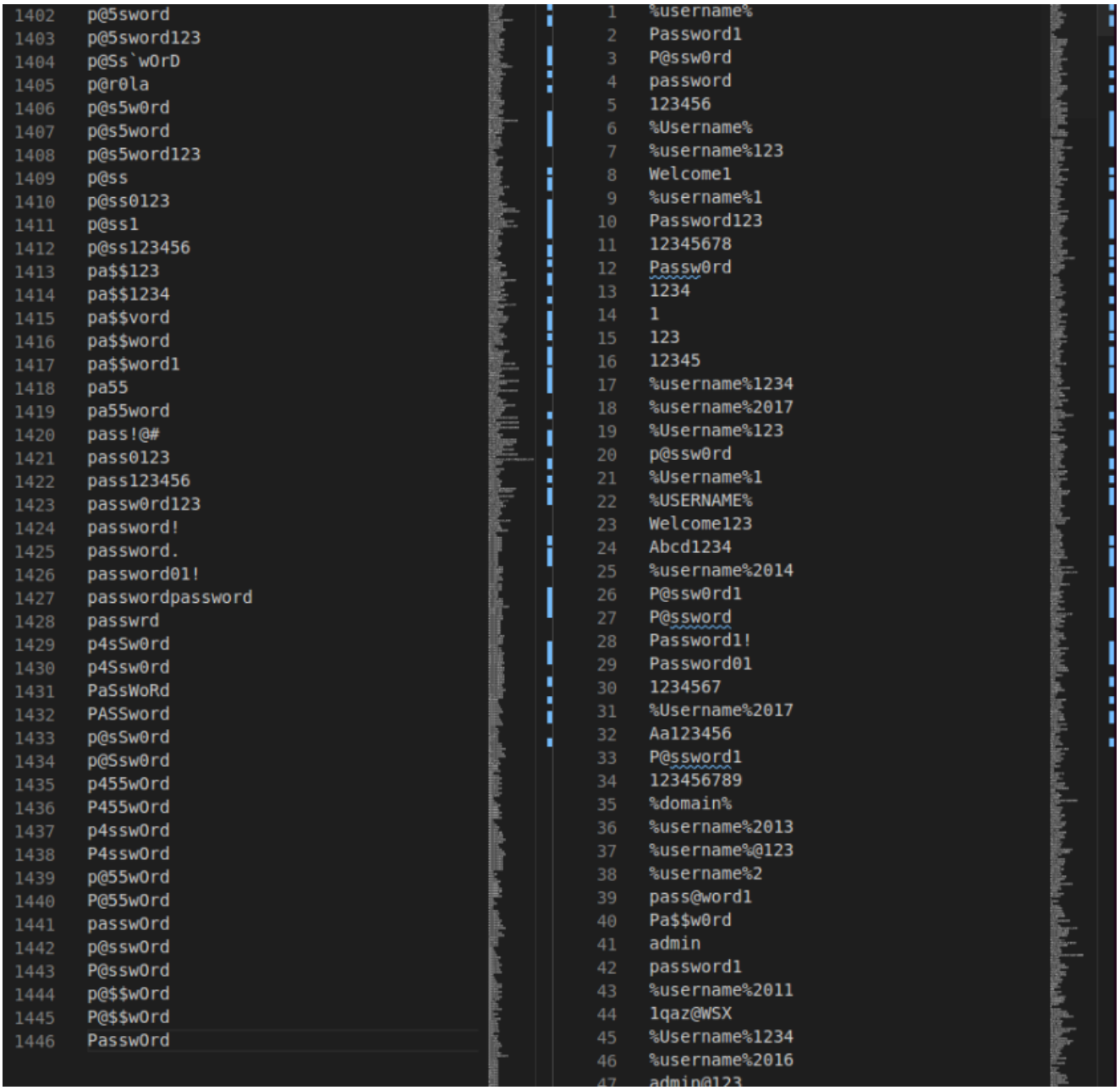
cyber.wtf/2020/08/31/trickbot-rdpscandll-password-transof/

August 31, 2020

After some weeks of not seeing the RDP scanner module of Trickbot, I recently observed that the module was again distributed among the bots in our tracking lab. Since Bitdefender already published a report on the module in March 2020, I focused on checking whether or not the command-and-control (C2) communication of the module remained more or less the same or if there was anything groundbreakingly new. Short answer: there wasn't. There may be some under-the-hood fixes or improvements but I (as of yet) did not stumble upon anything significant that wasn't already found by Bitdefender: the module still receives its mode of action, target servers, usernames, and password candidates from the C2 server and then does what the mode tells it to do. But while I was checking that, I also had a look at the actual data that we received from the C2 server.

Password List

My intuition on the password list was that it is just a dictionary of words to try. This is also suggested by the URL which is used to retrieve the password list: `hxxps://%c2%/%gtag%/%bot_id%/rdp/dict`. Thus I did not have a closer look at the password list at that time, because everything looked the way Bitdefender described it and I had no reason to look at it in detail. But one or two days later, I re-requested the list of passwords to see whether the list changed in the meantime – and it did indeed. Because of that I had a quick look at what changed and then I noticed that I overlooked something right from the start (literally, duh!). On the left side of the picture you see what I had a quick look at after retrieving the password list from the C2 server with curl (and thus seeing only the last lines of the output). On the right side there is the very same password list, just seen from the start.



To the keen eye it seems that they may be using some kind of templating mechanism to adjust the list of passwords and use more specific credential candidates. With that thought in mind I spun up my analysis environment and started digging into the module to see what the Trickbot gang is actually doing there (spoiler: yes, they do some kind of templating – but not just the find-and-replace kind).

Transforming them P@ssw0rds

As mentioned before, this is not a simple find-and-replace but instead they can change the credential candidates to better fit the attacked host. In that sense, I decided to call those things transforms instead of templates because they are not just templates that are filled out but a little bit more dynamic. Example:

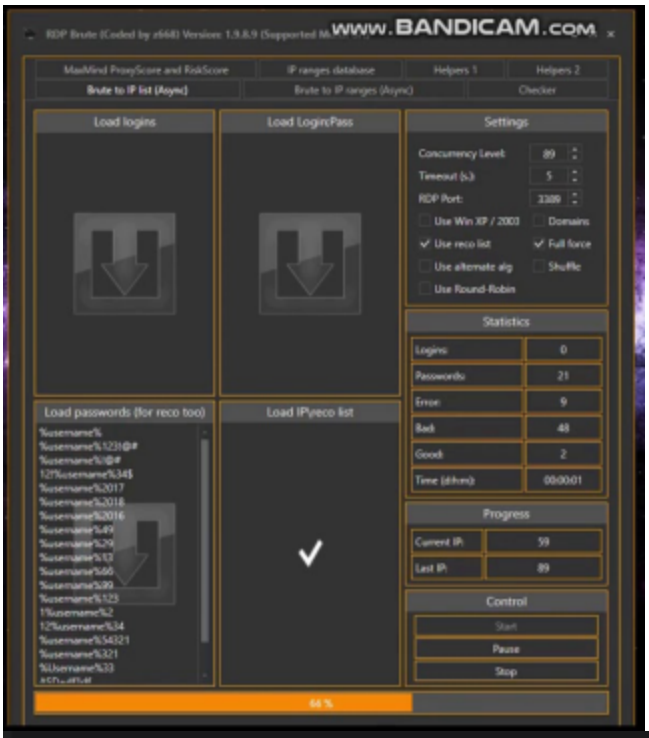
- `%username%123` → `myuser123` (lowercase)
- `%Username%123` → `Myuser123` (lowercase but first char uppercase)
- `%UsErNaMe%123` → `MyUsEr123` (alternating case, starting with uppercase char)
- `%EMANRESU%123` → `RESUYM123` (uppercase and reversed)

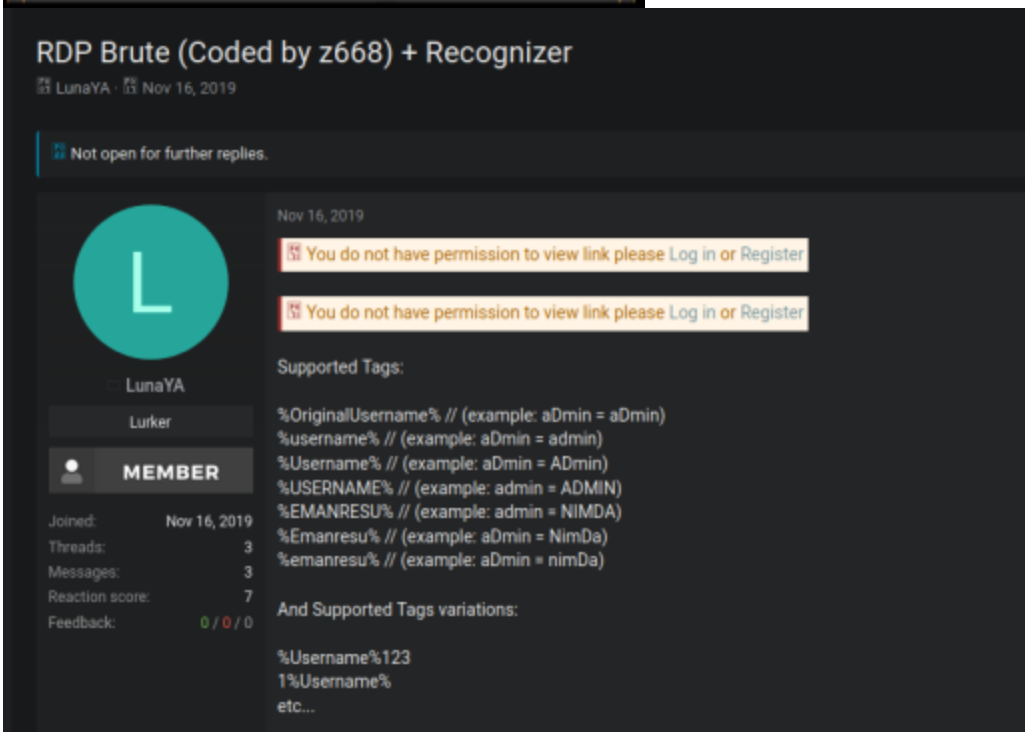
And that is essentially how the markers in the password list work. I was able to extract all 91 transformations that are currently available to the rdpscanDll (as of 2020-08-14). Please find the list with all transforms with an example and a description for each of them at the end of this blog post.

Some of the transforms can even be parameterized to a certain degree: %OriginalUsername%, %OriginalDomain%, and %domain% can be prepended or appended with an `(N)` to indicate whether the first N or last N characters of the element should be used (or everything if no parameters are present).

Reconnaissance

After finding the list of transforms, I decided to ask my favorite internet search engine whether these names for the transforms are known related to RDP. And I indeed found a RDP brute force tool by a certain z668 which seemingly makes use of some of the transforms that are used in the rdpscanDll. Although this tool seems to be a standalone application, the names of the transforms and the context of their use could suggest a connection between z668 and the Trickbot gang – at least to a certain degree. Sure, the connection may not be really strong because the Trickbot module is written in C++ and the RDP tool seems to be written in C#. But given the fact that C# can load and use native DLLs and considering that z668 forked the FreeRDP project on Github, the actual scanner may indeed be written in C/C++ (and probably using FreeRDP). Thus it is possible that the Trickbot gang may have obtained the source code from z668 to integrate the RDP scanner into their module framework and to use their C2 communication protocol. But: this is just guessing based on some more or less loose facts – I could easily be completely wrong with that.





Transform List

Transform Identifier	Example	Description
EmptyPass		tries an empty password

Transform Identifier	Example	Description
GetHost		fills in the hostname of the currently attacked IP (ex: myhost)
IP		the currently attacked IP address (ex: 234.234.234.234)
Port		fills in the currently attacked port (ex: 3389)
IpReplaceDot	234.234.234.234 → 234234234234	remove the dots of the IP address
RemoveNumerics	us3rn4me → usrnme	removes all number from the username
RemoveLetters	us3rn4m3 → 343	removes all letters from the username
RemoveOtherSymbols	usern@m3 → usernm3	removes all non-alphanumeric characters from the username
OriginalUsernameLettersBeginInverse	123admin456 → 123654nimda	keeps all non-letters (i.e. digits, special chars) at the beginning of the username and reverses the rest (“invert [from where] letters begin
OriginalUsernameLettersBeginSwap	123admin456 → admin456123	swaps all non-letters (i.e. digits, special chars) at the beginning of the username with the rest (“swap [where] letters begin”)

Transform Identifier	Example	Description
OriginalUsernameLettersEndInverse	admin123root → admintoor321	keeps all letters at the beginning of the username and reverses the rest (“invert [where] letters end”)
OriginalUsernameLettersEndSwap	admin123root → 123rootadmin	swaps all letters at the beginning of the username with the rest (“swap [where] letters end”)
OriginalUsernameNumsBeginInverse	admin123root → admintoor321	keeps all non-digits at the beginning of the username and reverses the rest (“invert [from where] nums begin
OriginalUsernameNumsBeginSwap	admin123root → admintoor321	swaps all non-digits at the beginning of the username with the rest (“swap [where] nums begin”)
OriginalUsernameNumsEndInverse	123admin → 123nimda	keeps all digits at the beginning of the username and reverses the rest (“invert [where] nums end”)
OriginalUsernameNumsEndSwap	123admin456 → admin456123	swaps all digits at the beginning of the username with the rest (“swap [where] nums end”)

Transform Identifier	Example	Description
OriginalUsernameInsert	%OriginalUsernameInsert% (N)SOMESTRING → SOMEusernameSTRING (ex: N = 4)	insert username after Nth character of SOMESTRING
OriginalUsername		use the username as password
OnlyName	Firstname Lastname → Firstname	uses only the first name (everything left of the first space) of the username as password
OnlySurname	Firstname Lastname → Lastname	uses only the last name (everything right of the first space) of the username as password
username	Admin → admin	username in lowercase
Username	AdMin → Admin	username lowercase but first char upper
UsErNaMe	Admin → AdMiN	username in alternating case, starting with uppercase
uSeRnAmE	Admin → aDmIn	username in alternating case, starting with lowercase
USERNAME	Admin → ADMIN	username in uppercase
EMANRESU	Admin → NIMDA	username in uppercase and reversed

Transform Identifier	Example	Description
EmanresuLowercase	AdMin → Nimda	username reversed and lowercase, first char uppercase
Emanresu	AdMin → NiMdA	username reversed, first char upper
emanresuLowercase	AdMin → nimda	username reversed and lowercase
emanresuUppercase	AdMin → NIMDA	username reversed and uppercase
emanresu	Admin → nimda	username reversed and lowercase
ReplaceFirst_X-x	administrator → @dministrator (ex: %ReplaceFirst_a-@%)	replaces the first occurrence of X with x in the username (needle and replacement can be more than 1 char)
ReplaceFirstI_X-x	Administrator → @dministrator (ex: %ReplaceFirstI_a-@%)	case insensitively replaces the first occurrence of X with x in the username (needle and replacement can be more than 1 char)
ReplaceLast_X-x	administrator → @dministrator (ex: %ReplaceLast_a-@%)	replaces the last occurrence of X with x in the username (needle and replacement can be more than 1 char)

Transform Identifier	Example	Description
ReplaceLastI_X-x	Administrator → @dministrator (ex: %ReplaceLastI_a-@%)	case insensitively replaces the last occurrence of X with x in the username (needle and replacement can be more than 1 char)
ReplaceAll_X-x	administrator → @dministrator (ex: %ReplaceAll_a-@%)	replaces all occurrences of X with x in the username (needle and replacement can be more than 1 char)
ReplaceAllI_X-x	Administrator → @dministrator (ex: %ReplaceAllI_a-@%)	case insensitively replaces all occurrences of X with x in the username (needle and replacement can be more than 1 char)
DomainRemoveNumerics	test-123.com → test-.com	removes all digits from the domain
DomainRemoveLetters	test-123.com → -123.	removes all letters from the domain
DomainRemoveOtherSymbols	test-123.com → test123com	removes all non-alphanum chars from the domain
OriginaldomainInsert	%OriginaldomainInsert% (N)SOMESTRING → SOMEdomainSTRING (ex: N = 4)	insert domain after Nth character of SOMESTRING

Transform Identifier	Example	Description
OriginaldomainPart	test-123.com → 123com (ex: %OriginaldomainPart%(6))	takes the last N chars of the domain name (ignoring any dots)
OriginaldomainNumsBeginInverse	test-123.com → test-moc.321	keeps all non-digits at the beginning of the domain and reverses the rest (“invert [from where] nums begin
OriginaldomainNumsBeginSwap	test-123.com → 123.comtest-	swaps all non-digits at the beginning of the domain with the rest (“swap [where] nums begin”)
OriginaldomainNumsEndInverse	123-test.com → 123moc.tset-	keeps all digits at the beginning of the domain and reverses the rest (“invert [where] nums end”)
OriginaldomainNumsEndSwap	123-test.com → -test.com123	swaps all digits at the beginning of the domain with the rest (“swap [where] nums end”)
OriginaldomainLettersBeginInverse	test-123.com → test-moc.321	keeps all non-letters (i.e. digits, special chars) at the beginning of the domain and reverses the rest (“invert [from where] letters begin

Transform Identifier	Example	Description
OriginaldomainLettersBeginSwap	123-test.com → test.com123-	swaps all non-letters (i.e. digits, special chars) at the beginning of the domain with the rest (“swap [where] letters begin”)
OriginaldomainLettersEndInverse	test-123.com → testmoc.321-	keeps all letters at the beginning of the domain and reverses the rest (“invert [where] letters end”)
OriginaldomainLettersEndSwap	test-123.com → -123.comtest	swaps all letters at the beginning of the domain with the rest (“swap [where] letters end”)
Originaldomainleft	test-123.com → test-123	takes the left part of the domain (everything left of the first dot) and lowercases the first character
OriginalDomainleft	test-123.com → Test-123	takes the left part of the domain (everything left of the first dot) and capitalizes the first character
Originaldomainright	test-123.com → test-123	takes the right part of the domain (everything right of the first dot) and lowercases the first character

Transform Identifier	Example	Description
OriginalDomainright	test-123.com → Test-123	takes the right part of the domain (everything right of the first dot) and capitalizes the first character
Originaldomain		uses the plain domain name
OriginalDomain	test-123.com → Test-123.com	uses the domain name and capitalizes the first character
NiamodLowercase	abc%NiamodLowercase%123	abc123
niamodLowercase	test-123.com → Moc.321-tset	reverses and lowercases the domain name, first character capitalized
niamodUppercase	test-123.com → mOC.312-TSET	reverses and capitalizes the domain name, first char lowercase
domainleftHyphen	test-123.com → test	takes everything left of the first hyphen
DOMAINLEFTHYPHEN	test-123.com → TEST	takes everything left of the first hyphen, capitalized
DomainleftHyphen	test-123.com → Test	takes everything left of the first hyphen, first char capitalized
domainrightHyphen	test-123.com → 123.com	takes everything right of the first hyphen

Transform Identifier	Example	Description
DOMAINRIGHTHYPHEN	test-123.com → 123.COM	takes everything right of the first hyphen, capitalized
DomainrightHyphen	test-abc.com → Abc.com	takes everything right of the first hyphen, first char capitalized
domainleftUnderscore	test_123.com → test	takes everything left of the first underscore
DOMAINLEFTUNDERSCORE	test_123.com → TEST	takes everything left of the first underscore, capitalized
DomainleftUnderscore	test_123.com → Test	takes everything left of the first underscore, first char capitalized
domainrightUnderscore	test_abc.com → abc.com	takes everything right of the first underscore
DOMAINRIGHTUNDERSCORE	test_123.com → 123.COM	takes everything right of the first underscore, capitalized
DomainrightUnderscore	test_abc.com → Abc.com	takes everything right of the first underscore, first char capitalized
DomainReplaceFirst_X-x	EXAMPLE-attack.com → EXAMPLE-@ttack.com (ex: %DomainReplaceFirst_a-@%)	replaces the first occurrence of X with x in the domain (needle and replacement can be more than 1 char)

Transform Identifier	Example	Description
DomainReplaceFirstI_X-x	EXAMPLE-attack.com → EX@MPLE-attack.com (ex: %DomainReplaceFirstI_a-@%)	case insensitively replaces the first occurrence of X with x in the domain (needle and replacement can be more than 1 char)
DomainReplaceLast_X-x	EXAMPLE-attack.com → EXAMPLE-att@ck.com (ex: %DomainReplaceLast_a-@%)	replaces the last occurrence of X with x in the domain (needle and replacement can be more than 1 char)
DomainReplaceLastI_X-x	EXAMPLE-attack.com → EXAMPLE-att@ck.com (ex: %DomainReplaceLastI_a-@%)	case insensitively replaces the last occurrence of X with x in the domain (needle and replacement can be more than 1 char)
DomainReplaceAll_X-x	EXAMPLE-attack.com → EXAMPLE-@tt@ck.com (ex: %DomainReplaceAll_a-@%)	replaces all occurrences of X with x in the domain (needle and replacement can be more than 1 char)
DomainReplaceAllI_X-x	EXAMPLE-attack.com → EX@MPLE-@tt@ck.com (ex: %DomainReplaceAllI_a-@%)	case insensitively replaces all occurrences of X with x in the domain (needle and replacement can be more than 1 char)
niamod	test-123.com → moc.321-tset	reverses the domain name

Transform Identifier	Example	Description
Niamod	test-123.com → Moc.321-tset	reverses the domain name, first char capitalized
domainleft	TEST-123.com → test-123	everything left of the first dot, lowercased
DOMAINLEFT	Test-123.com → TEST-123	everything left of the first dor, capitalized
Domainleft	test-123.com → Test-123	everything left of the first dot, lowercased but first char capitalized
domainright	TEST-123.com → com	everything right of the first dot, lowercased
DOMAINRIGHT	Test-123.com → COM	everything right of the first dor, capitalized
Domainright	test-123.com → Com	everything right of the first dot, lowercased but first char capitalized
domain	TEST-123.com → test-123.com	domain name, lowercase
Domain	TEST-123.com	domain name lowercased, first char capitalized
DoMaIn	test-123.com → TeSt-123.cOm	domain name in alternating case, starting with uppercase
dOmAiN	test-123.com → tEsT-123.CoM	domain name in alternating case, starting with lowercase

Transform Identifier	Example	Description
DOMAIN	test-123.com → TEST-123.COM	domain name capitalized
NIAMOD	test-123.com → MOC.321-TSET	domain name reversed and capitalized