Engagement 4 Intended Vulnerabilities

The Apogee Research effort on STAC is funded by the United States Airforce Research Lab (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract Number FA8750-15-C-0089. THE VIEWS AND CONCLUSIONS CONTAINED IN THIS DOCUMENT ARE THOSE OF THE AUTHORS AND SHOULD NOT BE INTERPRETED AS REPRESENTING THE OFFICIAL POLICIES, EITHER EXPRESSED OR IMPLIED, OF THE DEFENSE ADVANCED RESEARCH PROJECTS AGENCY OR THE US GOVERNMENT.

Challenge Overview: AirPlan

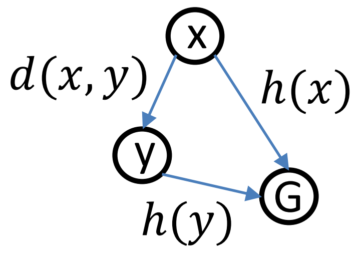
- **Description**: Airplan is a web-based service for airlines. It allows users to upload graphs representing flight route plans and analyzes the flight routes and crew-scheduling.
- **Vulnerability:** AC Time
  - Ford Fulkerson max flow problem with rational capacities
  - A* search for shortest path with an inconsistent heuristic
  - Dijkstra's shortest path where negative weights are allowed
  - Merge sort where n%6=0 results in unnecessary merges
- **Vulnerability:** AC Space
  - XML bomb on input graph
- **Vulnerability:** SC Space
  - Padding on response to display graph request allows mapping of number of airports in flight plan to packet size. Operational Budget: 2 operations; 100% probability of success
  - Packet size maps to whether or not a given graph is connected or disconnected. Operational Budget: 2 operations; 100% probability of success

Additional Detail: Max Flow (AC Time)

- Final AirPlan delivery expected to allow users to create a flight schedule with a limited number of crews
- Given an input graph G that represents a set of flights F and k flight crews, the current AirPlan delivery contains a method which uses the Ford Fulkerson algorithm to determine whether the it is possible to construct a flight plan to service all the flights in F
  - Given an input graph G, capacity c, source node s, and sink node t Ford Fulkerson calculates the max flow from s to t
  - Given a graph with E edges and a max flow f, assuming integer capacity for each edge Ford Fulkerson complexity is bounded by $O(fE)$
  - With rational capacities terminates with different complexity bounds

Additional Detail: A* Shortest Path (AC Time)

- Given that a user has uploaded a graph, AirPlan allows the user to search for the shortest flight between two airports using the A* search.
- A* search – find the minimum cost path from source to destination.
  - The decision on which node to expand at a given point is controlled by $f(n) = g(n) + h(n)$ where $g(n)$ is the cost from source to n and $h(n)$ is the estimated cost from n to the destination
  - The performance of A* is dependent on the heuristic function $h(n)$. The performance of $h(n)$ is defined by:
    - Admissible – doesn't overestimate cost; $h(n) \leq h^*(n)$ where $h^*(n)$ is true cost
    - Dominance – given 2 admissible heuristics $h_1(n)$ and $h_2(n)$ if $h_2(n) \geq h_1(n)$ then $h_2(n)$ expansions $\leq h_1(n)$ expansions
    - Consistency – $h(x) \leq d(x,y) + h(y)$
- Vulnerable AirPlan uses inconsistent, admissible heuristic results in $O(2^N)$ expansions
- This intended AC time vulnerability can cause an AC Space vulnerability due to the unnecessary node expansions



Additional Detail: Dijkstra’s Shortest Path (AC Time)

- Given that a user has uploaded a graph, AirPlan allows the user to search for the shortest flight between two airports using Dijkstra's Algorithm.
- Dijkstra's Algorithm – find the minimum cost path from source S to destination D.
  - The decision of which node to expand is controlled by $g(n)$ where $g(n)$ is the cost from S to node n.
  - Initial: $g(S) = 0$ and $g(n) = \infty$. Cost from S to n is $w_{s,n}$
    - Begin at S and for each non-visited neighbor S_n of S calculate $g(S_n) = g(S) + w_{s,n}$
    - If $g(S_n) < g(n)$ set $g(n) = g(S_n)$
    - Mark S as visited and proceed to the non-visited node with the least cost
    - Proceed until D is found
- For non negative costs, runtime is $O(|V|^2)$ where V is number of nodes
- Vulnerable algorithm allows negative costs. Input graphs with cycles with negative costs are non-terminating

Additional Detail: Space Vulnerabilities

- AC Space
  - Airplan allows users to input flight plans as XML files. Each vertex represents an airport and each edge represents a flight.
  - The XMLFileLoader class which parses the XML input expects each line of the XML file to represent a single graph entity (vertex or edge)
  - Users can input a graph with nodes which reference predefined nodes and cause the challenge application to exceed the resource usage limit
  - This intended AC Space vulnerability can cause an AC Time vulnerability
- SC Space Number of Airports
  - When a user inputs a graph, after the server processes the graph, it redirects the user to a page where the graph is displayed
  - Airplan displays user graphs with a table template which leaks the number of vertices (cities) in a given graph
- SC Space Connected Graph
  - Airplan reports the properties of a given graph is a special packet. Each component of the packet except the connected graph component has a fixed size
  - The resulting size of the component packet leaks whether or not a graph is connected

Summary of BidPal

- **Description**: Bid Pal is a peer-to-peer application that allows users to buy and sell items via a highest-bidder-wins auction.
- **Vulnerability**: SC Time (Same as PowerBroker)
  - Secret: A user’s bid
  - Max Number of Operations: 2
  - Probability of Success: 75%
- **Vulnerability**: AC Time
  - Input Budget: 800,000
  - Resource Usage Limit: 1200 seconds

Additional Detail: Checksum (AC Time)

- Following the creation of a bid comparison message, checks are performed to check the validity of the username and auction ID
- If the username contains "NO_USER" and the auction ID contains "NO_AUCTION_ID", the validity checks fail and repeated comparison messages are created resulting in AC vulnerability.

Summary of Collab

- **Description**: Collab is a scheduling application which allows users to create new events and modify existing events
  - Each user’s event is identified by a unique event ID which represents the number of time units since the application’s epoch
  - The application contains a special class of users called auditors who can schedule special auditing events for other users
  - The audit events and their event ID's are hidden from normal users
  - The application contains a built-in user ‘picard’ who has an audit event scheduled
- **Initial Vulnerability**: SC Time
  - Secret: the event ID for picard’s audit event
  - Max Number of Operations: 200 operations (139 required)
  - Probability of Success: 90%
- **Null AC Time**
  - Input Budget: 5000 bytes
  - Resource Usage Limit: 90 seconds

Intended SC Time Vulnerability

- The challenge program creates a temporary calendar when a user logs in.
  - All public events are copied to the temporary calendar
  - Each node covers a period of time (variable based on calendar entries).
  - If more than 8 elements in a node, node is split into two nodes (taking an extra 639 µs over a simple event addition)
  - After public data structure is built, Audit events are added in.
- Summary of Attack Strategy:
  - Attacker can predict the data structure from the public events
  - Add public events in relevant time period until there should be 8 entries
  - Measure time for each add – if any are long, then there are audit events
  - Refine Audit ID (start time of audit) by binary search in time period
- Operational Budget
  - **Max number of operations**: 200 operations (139 required)
  - **Probability of success**: 90%

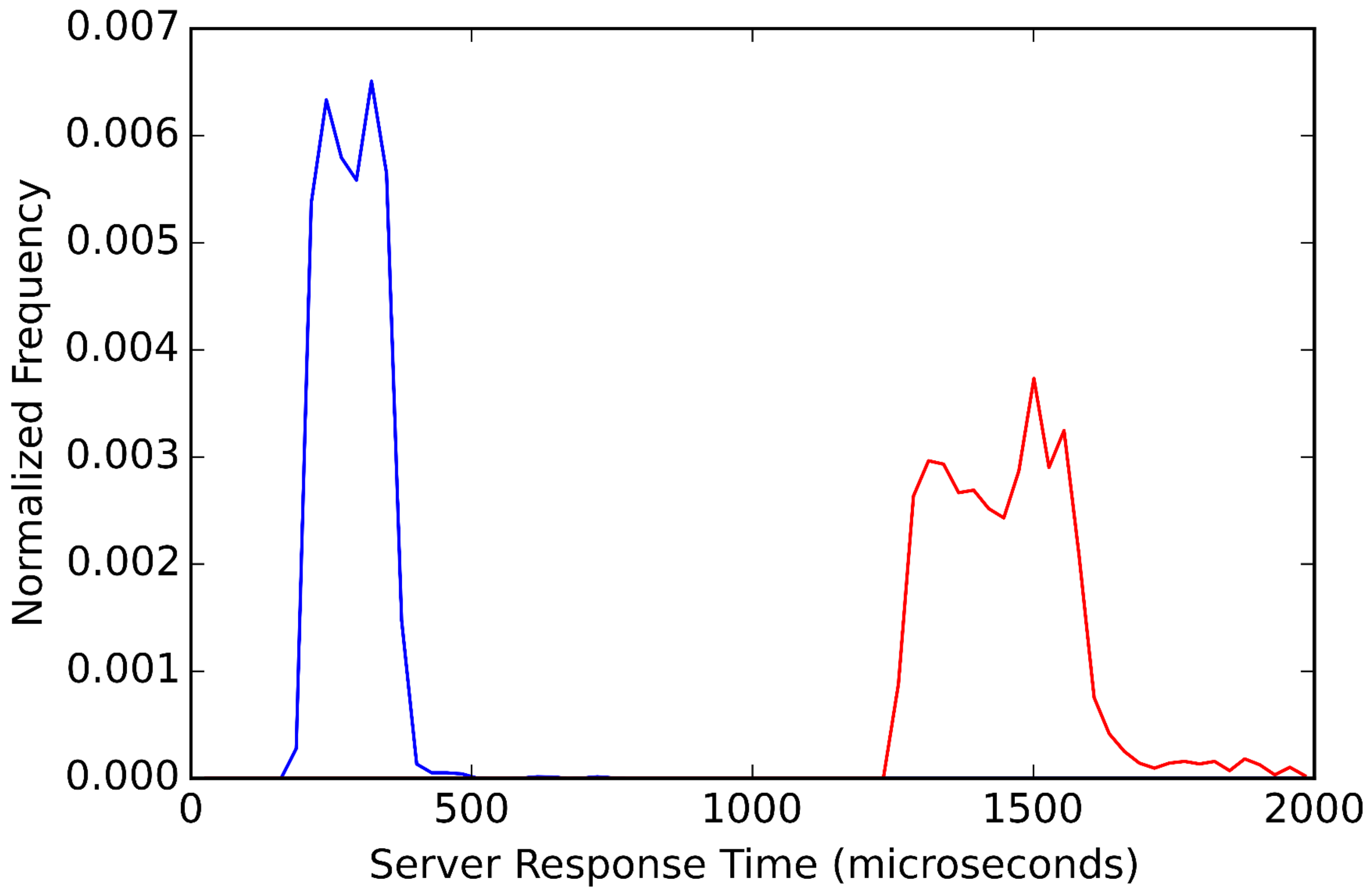
Removing SC Time Vulnerability

- Side channel caused by time difference between splitting and non-splitting inserts
  - BBN strengthened side channel by logging actions in the case of splitting inserts
- Path from User Input to vulnerability
  - CollabServer.channelRead0 → SchedulingSandbox.add() → SchedulingSandbox.addhelper() ↓

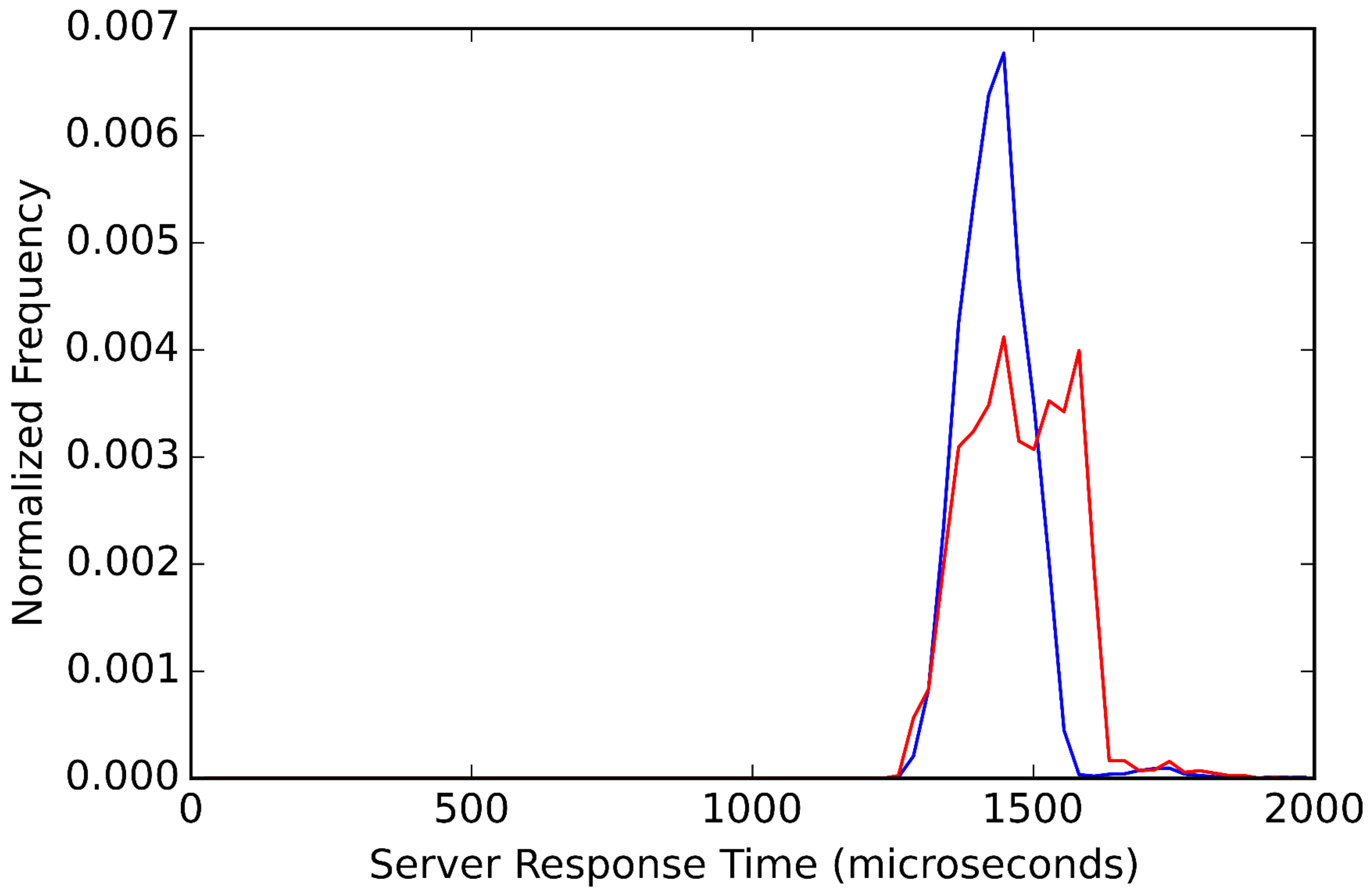
```
if(eventidlist[ind]=ENDVAL) //indicates node full
       split() //uses 3 log writes to amplify SC
```

- Non-locally balanced non-split inserts to weaken side channel
  - Added balancing operation after SchedulingSandbox.add() method call
  - Independent analysis of SchedulingSandbox.add() or SchedulingSandbox.addhelper() methods show strong side channel
  - Complete analysis of CollabServer.channelRead0 method shows weak side channel

Vulnerable Insert Operation



Non Vulnerable Insert Operation



Summary of InfoTrader

- **Description**: InfoTrader is a document server for stock trading professionals. Users can view the sitemap of stored documents, insert a document, or retrieve a document. Documents can contain links to existing InfoTrader documents. After a new document is uploaded InfoTrader regenerates and updates the sitemap file.
- **Vulnerability:** AC Space
  - Input Budget: 5000 bytes
  - Resource Usage Limit: 100 MB (119.35 MB observed)
- **Null:** AC Time
  - Input Budget: 5000 bytes
  - Resource Usage Limit: 1000 seconds

Additional Detail : AC Vulnerability

- Sitemap generation method is vulnerable
  - Users can upload documents containing links to other documents
  - When InfoTrader generates the sitemap it runs a modified DFS and follows all links to generate the sitemap
  - The modified DFS contains input guards to prevent AC Time (cycling) and AC Space (unnecessary traversal) vulnerabilities
  - The input guard is "broken"; allows users to upload a document with the an empty string, which happens to be the tag for the root node of the sitemap, and isn't blocked
- AC Space exploit:
  - Attacker can upload an exploit document with same name as root node
  - Attacker can use input budget to upload documents which link to the exploit document
  - During sitemap generation given *n* documents linking to the exploit document, the resulting sitemap will contain *y* traversals of all the documents in InfoTrader where $y = \sum_{k=0}^{n} \frac{n!}{k!}$
- Null AC Time vulnerability
  - Input guard appears vulnerable to AC Time vulnerability but an error occurs and is gracefully handled

Summary of Linear Algebra Platform

- **Description**: The Linear Algebra Platform is a service that allows for multi-core computation of linear algebra operations (matrix multiplication, shortest path, Laplacian, minimum spanning tree). The application uses a mixture of multi-core and single-core algorithms to compute solutions.
- **Vulnerability**: AC Time
  - The matrix multiplication operation of the application is vulnerable
  - Input Budget: 15,250,000 bytes (Header + 2, 650x650 Matrices *)
  - Resource Usage Limit: 225 seconds (≥ 229 seconds observed)
- **AC Space**
  - Input Budget: 15,250,000 bytes
  - Resource Usage Limit: 2500 MB (> 4000 MB observed)

Additional Detail: Supported Operations

- Linear Algebra Platform supported operations:
  - Matrix Multiplication: Computes multiplication of 2 N x N matrices (Vulnerable)
  - Multi point Shortest Path: Given k vertices, with k<log(N) and an N x N matrix, computes the shortest path length from the k vertices to every other vertex in the matrix using a binary heap implementation of Dijkstra's Algorithm (Not Vulnerable)
  - Graph Laplacian: Computes the Laplacian of a graph given its N x N adjacency matrix (Not Vulnerable)
  - Minimum Spanning Tree: Computes the minimum spanning tree of a weighted non-directed graph given its N x N adjacency matrix (Not Vulnerable)
- The application enforces $N \leq 1000$ for all input matrices
  - We believe that BBN should either reduce the enforced matrix size or we should set the budget on the enforced matrix size.
  - The latter will point the blue teams to matrix multiple, as that will be the only function that is “large”

Additional Detail: AC Time Vulnerability

- The Matrix Multiplication operation of the Linear Algebra Platform is vulnerable
  - The application receives two matrices A and B
  - The application calculates the mean row skew of matrix A
    - Fast Case: If ABS(mean row skew A) < ln(2), dot product operations are split between two threads evenly. (N = 650, **runtime**: 106 to 182.9 seconds)
    - Slow Case: If ABS(mean row skew A) >= ln(2), dot product operations only done on one thread. (N = 650, **runtime**: 229 to 260 seconds)
  - **Input Budget**: 15,250,000 bytes (N = 650 for Matrix Multiplication)
  - **Resource Usage Limit**: 225 seconds
- The Multi-point shortest path operation appears weakly vulnerable by STAC definition
  - The operation uses two threads to compute the shortest path
  - The **number of tasks on each thread is random** and not controlled by user input
    - Should **non-user controlled non-deterministic complexities** be allowed?
- The complexities of all the non-vulnerable operations are less than the complexity of the matrix multiplication operation

Additional Detail: AC Space Vulnerability

- The Shortest path operation of the Linear Algebra Platform is vulnerable
  - The application receives two matrices an NxN adjacency matrix and a vector of k vertices.
  - The application allocates memory for the vector of k vertices before checking its size.
- Attacker can submit a $0\ x\ 0$ targetNodes matrix and claim that it is an $M\ x\ M$ matrix where M is a very large number. This will trigger an illegal argument exception but not before the resource usage limit is exceeded.

Summary of Malware Analyzer

- **Description**: Malware Analyzer is a server application that helps analysts better understand malware. The application contains two malware classifiers, a packer detector, and a control flow graph generator. Both classifiers compute a pairwise similarity metric between different malware and return a rank ordered list of the top 5 most similar malware for a given input malware.
- **Vulnerability:** AC Space
  - Logging feature contains AC Space vulnerability
  - Input Budget: 3,000 bytes
  - Resource Usage Limit: 80,000 bytes (92,564 observed)
- **Null:** AC Time
  - Input Budget: 3,000 bytes
  - Resource Usage Limit: 30 seconds (<2 seconds observed)

Additional Detail: AC Space Vulnerability

- Malware analyzer uses a log file to track events that occur when a user submits new malware for classification. In benign cases the log file **overwritten** with information about the most recent malware submitted for classification (**bounded size**).
- When an error occurs during classification a malicious user can cause the entire stack trace of the error to be logged. This error is **appended** therefore repeated erroneous requests can cause the log file to **monotonically increase in size**.
- Malware Analyzer uses feature vectors, f_v, of 256 integers to describe malware. The packer detector determines whether malware is packed or unpacked based on heuristic properties of its feature vector. The calculation of the packed heuristic causes the AC Space vulnerability.
  - The average byte value heuristic calculates: $\frac{\sum_{index=0}^{255} index * f_v[index]}{s = \sum_{index=0}^{255} f_v[index]}$
  - When $s = 0$ an arithmetic exception is thrown and the resulting stack trace is appended to the log file; s is an integer:
    - s rolls over at $2^{31} - 1$ to -2^{31}
    - s rolls over at -2^{31} to 0
  - Vulnerability is triggered for feature vectors where the sum of all elements is $(2^{31} - 1) + 1 + 2^{31} = 4{,}294{,}967{,}296$

Summary of PowerBroker

- **Description:** PowerBroker is a peer-to-peer program used by power suppliers for exchanging power. Suppliers with excess power attempt to auction off power; those with deficits attempt to purchase power. The application uses a 1024-bit RSA key to secure communications and a secure multiparty computation protocol to compare bids.
- **Vulnerability:** SC Time
  - Secret: A user’s private key
  - Max Number of Operations: 75,000
  - Probability of Success: 80%
- **Vulnerability:** SC Time
  - Secret: A user’s bid
  - Max Number of Operations: 2
  - Probability of Success: 75%
- **Vulnerability:** AC Space
  - Input Budget: 800,000 bytes
  - Resource Usage Limit: 1,048,576 bytes

Additional Detail: RSA SC Time

- PowerBroker uses a 1024-bit RSA key to encrypt communications between users (power auction participants).
- User A can issue a challenge to user B who must then decrypt the challenge to verify B's identity to A.
- The PowerBroker RSA protocol uses the Chinese Remainder Theorem (CRT) and Montgomery Reductions to increase the speed of the modular exponentiation in decryption.
- The SC Time vulnerability results in a Sawtooth function that maps input values to the runtime of the decryption.
- If a sequence of inputs representing increasing integers is supplied, the period of the Sawtooth function is the private key of the user performing the decryption.

Additional Detail: Bid Amount SC Time

- A single auction in the PowerBroker challenge has one user acting as an auctioneer and all other users acting as bidders (including the auctioneer)
- When the auction process begins all users submit a single bid to all other users
- Each user compares their individual bid with the bids from all other users and determines whether their bid was higher, lower, or the same as each of the other bids
- The process of comparing bids uses a secure multiparty computation protocol intended to obscure a user's bid from all other users
- The bid comparison method contains an SC Time vulnerability that allows an adversary to determine a users bid to within $30 with an 80% probability of success
- A single auction only contains one bidding cycle therefore an adversary cannot change their bid during the bidding cycle after learning another user's bid
- An adversary can use the information to determine a user's bidding strategy in order to win future auctions

Additional Detail: Logging AC Space

- The PowerBroker connection protocol contains a contingency which attempts to account for interruptions during the establishment of a connection between two users
- When a user A attempts to establish a connection with user B, if there is an interruption, user B will attempt to reconnect with user A
- Each user has a log file which records each attempt to re-establish a connection
- An adversary can connect to a victim and disconnect from the victim. The victim will continuously attempt to reconnect logging each attempt in his/her log file.
- The adversary can cause the victim to exceed the resource usage limit

Summary of RSA Commander

- **Description**: RSA Commander is a secure messaging application that uses a combination of DES and RSA to encrypt messages. Each user has a separate running instance of the application bound to a port on a given host, users can send messages specifying the host and port of the destination user.
- **Vulnerability:** AC Time
  - The RSA encryption contains an AC Time vulnerability
  - Input Budget: 5000 bytes
  - Resource Usage Limit: 2 hours (3 hours 47 minutes observed)
- **Null:** SC Time
  - Secret: plaintext of message intended for another user
  - Max Number of Operations: 10,000
  - Probability of Success: 80 %

Additional Detail: AC Time Vulnerability

- RSA Commander allows users to send and receive messages from other users. The application uses one thread to handle sending messages and a separate thread to handle receiving messages.
- The application contains an AC Time vulnerability that causes high CPU consumption on the victim's message receiving thread. This prevents the victim from receiving message during the attack.
- RSA commander uses a combination of DES and RSA to encrypt communication between two users.
  - DES – A random counter variable is encrypted via DES using a random nonce value as the key. The resultant value is XORed with the plaintext message to create cyphertext. The nonce and the counter are required to decrypt the cyphertext.
  - RSA – The nonce and counter are encrypted using RSA. The message sender chooses both values and encrypts them with the recipient's public key.
  - The recipient is sent a packet containing the RSA encrypted nonce and counter and the DES encrypted message cyphertext.
- AC Time vulnerability: RSA commander has slow and fast modes of RSA encryption

Additional Detail: AC Time Vulnerability

Messaging Protocol: Attacker Victim

Handshake open →

← Handshake response

Message →

←Acknowledge

Terminate→

- RSA Commander implements RSA encryption computation
  - $B^E mod(M)$ as $B.pow(E).mod(M)$; Runtime $= 0.578\,(\log_{10} B)^{1.9982}$ seconds
  - The computation is run for B = nonce (and other things that don't matter here)
  - N.B., there are other paths through the code for which this isn't the case, but sender can force this path to be taken (by setting counter = 0)
- Guard checks
  - $E \in [3, 65537]$, $M \in [0, 2^{512} - 1]$, nonce $< M$
  - Response increments nonce by 1; if nonce $= 2^{63} - 1$, roll nonce to zero
  - If the starting nonce $\geq 2^{63}$, then nonce doesn't roll and computation becomes expensive

Summary of SmartMail

- **Description:** SmartMail is an email application which allows users to send and receive emails from other users. SmartMail allows users query the application for the members of each mailing list and to send emails the entire mailing list. The email address of each mailing list administrator is a secret.
- **Vulnerability:** SC Space and Time
  - Secret: The email address of any mailing list administrator
  - Max Number of Operations: 200 (50 required)
  - Probability of Success: 100%
- **Null:** AC Time
  - Input Budget: 10,000 Bytes
  - Resource Usage Limit: 30 seconds (<1.1 seconds observed)
- **Null**: AC Space
  - Input Budget: 10,000 Bytes
  - Resource Usage Limit: 1024 MB (<300 MB observed)

Additional Detail: SC Space and Time

- Emails to mailing lists are sent to list members and administrator
- When the application is ready to send an email the list of recipients is sorted alphabetically, encrypted, and logged sequentially
  - Secret recipients (administrators) are logged in the secret log
  - Public recipients are logged in the public log
  - Non-existent recipients are logged in the public log
- Attacker can use log file write times to determine administrator email one character at a time
- Required Observables:
  - Attacker needs to be able to observe the log files
  - Attacker needs to be able to download program on server machine to track write time of each line in log file (ms resolution)

Summary of Tour Planner

- **Description**: Tour planner provides users with an efficient route through a set of requested cities. The first city requested by the user is the first city returned in the route.
- **Vulnerability**: SC Time
  - Secret: The cities requested by a user
  - Max Number of Operations: 300 operations (212 required)
  - Probability of Success: 95%
- **Null: SC Space**
  - Secret: The cities requested by a user
  - Max Number of Operations: 3000 operations (24,697 required)
  - Probability of Success: 100%
- **Null: AC Time**
  - Input Budget: 500 bytes
  - Resource Usage Limit: 30 seconds (<3 second observed)

Additional Detail: SC Time Vulnerability

- Tour Planner provides users with an efficient route through a set or requested cities.
  - First city requested by the user is the first city returned in the route
  - Routes are calculated from minimum spanning trees (MSTs) using Christofide's algorithm
  - MSTs are calculated using Prim's algorithm
- Side channel exists due to how tour planner reports its progress to a user
  - Each time a new edge is added to the MST, a message (packet is sent to the user)
  - Edges are added by traversing a sorted list (list is constant, arranged shortest to longest)
  - The list is traversed each time until the relevant edge is located thus there are regular timing differences between packets for an edge based on its location in the list
    - The timing difference between packets is effectively the amount of time it takes the program to traverse the ordered list to the relevant edge it is adding to the MST
    - Guessing the MST is really the goal—provides all of the necessary information
    - There are a finite number of MSTs that exist given a number of cities searched

Additional Detail: SC Time Algorithms

- At least two viable algorithms for guessing MSTs:
  - Euclidean Distance Method:
    - Represent each possible MST as a sorted vector of edge times
    - Calculate Euclidean distance between candidate times and each possible MST
    - Rank possible MSTs by Euclidean distance
    - **Worst Case Given 5 Cities**: 212 oracle queries for 95% probability of success
  - Mahalanobis Distance Method:
    - Represent each possible MST as n-dimensional Gaussian model
    - Calculate Mahalanobis distance between candidate times and MST Gaussian models.
    - Rank possible MSTs by Mahalanobis distance
    - **Worst Case Given 5 Cities**: 167 oracle queries for 95% probability of success
- Operational Budget for SC Time question:
  - **Max number of operation**: 300 operations
  - **Probability of success**: 95%

Additional Detail: Null SC Space

- Tour Planner encrypts traffic to remove the trivial side channel
- The encryption uses an AES256 cypher with a block size of 128 bits (16 bytes)
  - Result is that the payload traffic is binned into 16 byte bins
  - What is the strength of the resulting side channel?

Input (bytes)	Final Output (bytes)	# of Collisions	Entropy $\boldsymbol{log_2(\#col)}$
234	405	12887	13.65
218	405	24696	14.59
234	421	15414	13.91
218	389	132	7.04
250	421	1	0

- With an operational budget of 3000, the side channel is too weak

Challenge Overview: Tweeter

- **Description**: The Tweeter (formerly spell corrector) application is a clone of twitter (**is this a concern for DARPA?**). Users can create a profile, post tweets using hashtags, and see tweets posted by other users.
- **Vulnerability**: AC Time
  - The spell correction feature in the application is vulnerable
  - Input Budget: 2000 bytes (sign into the application and send a 140 character tweet)
  - Resource Usage Limit: 1500 seconds (≥ 2019 seconds observed)
- **Vulnerability**: AC Space
  - A deprecated setting which enables users to chose the size of their profile images is vulnerable
  - Input Budget: 2,500,000 bytes (create new 2048 users and specify the size of each user’s profile image)
  - Resource Usage Limit: 1500 MB (2028 MB observed)

Additional Detail: AC Time Vulnerability

- When a user posts a tweet, prior to being posted the server checks the tweet for spelling errors.
- If the application cannot find an error it immediately posts the message. Otherwise, show user correction options. The message posted after user input is provided.
- The spell corrector receives a word as an input argument
  - If the word is too long (>26 chars) or correct then the spell corrector returns null.
  - Spell check algorithm is a function of both
    - Word Length
    - Number of errors
    - n.b., the system will only consider a maximum number of errors before giving up
- Fast: A tweet with all correct words or words that are too long (**Runtime**: <2 seconds)
- Normal: A tweet with average length words containing few mistakes (**Runtime**: 10 to 262 seconds)
- Slow: A tweet with long (26 character) possible words multiple containing mistakes (**Runtime**: 2019 to 4079 seconds)

Additional Detail: AC Space Vulnerability

- The Tweeter application selects profile images for each user. The default size for each profile image is 128 x 128 pixels.
- Vulnerability Based on Deprecated Functionality:
  - A deprecated setting allows users to set the size of their profile images
  - This setting is not present in the normal interaction of users
  - Server still accepts the size setting as a valid input when a new profile is created
- The three options for file sizes are 128 x 128, 256 x 256, and 512 x 512
- AC Space:
  - Files size value: 128 (Memory Usage: 451,000 kB)
  - Files size value: 256 (Memory Usage: 737,916 kB)
  - Files size value: 512 (Memory Usage: 1,980,172 kB)
  - All inputs are the same size (just changing the value, not supplying the image)
- **Input Budget**: 2,500,000 bytes
- **Resource Usage Limit**: 1500 MB of memory

Summary of WithMi

- **Description:** WithMi is a peer-to-peer text chat and file transfer program. Users can connect to one another, create and join multi-party chat rooms, and share files. The applications uses a 1024-bit RSA key to secure communications and maintains a list of previous connections for each user to provide added security.

- **Vulnerability:** SC Time (Same as before)
  - Secret: A user's private key (RSA)
  - Max Number of Operations: 75,000
  - Probability of Success: 80%
- **Vulnerability:** SC Time
  - Secret: Whether two users have been previously connected (User Management)
  - Max Number of Operations: 1
  - Probability of Success: 80%
- **Vulnerability:** SC Space
  - Secret: Files shared between users
- **Vulnerability:** SC Space
  - Secret: Number of users in a chat

- **Vulnerability:** AC Time
  - Vulnerable Huffman decompression
  - Input Budget: 800,000 bytes
  - Resource Usage Limit: 210 seconds
- **Vulnerability:** AC Space
  - Unhandled negative integer in RLE decompression
  - Input Budget: 800,000 bytes
  - Resource Usage Limit: 120 MB
- **Vulnerability:** AC Space
  - Unhandled integer overflow in RLE decompression
  - Input Budget: 800,000 bytes
  - Resource Usage Limit: 120 MB

Additional Detail: SC Time - User Management

- When two WithMi users attempt to establish a secure connection, they exchange public keys and aliases
- When a user receives a connection request, the user checks his/her list of previous connection
  - If the connection is new, the user will add the new connection to the list of previous connections
  - Else the user will attempt to verify whether this connection is the same as the one stored in the list of previous connections
- Re-establishing a connection takes longer than establishing a new connection
- This side channel allows an adversary to determine whether two users have connected previously

Additional Detail: AC Vulnerabilities

- WithMi allows users to share files. WithMi accepts several compression formats as valid inputs. Restrictions are placed on the input file sizes for all files, however different variants of WithMi contain vulnerable decompression methods.
- **Huffman Decompression** AC Time
  - The Huffman Decompression method contains a vulnerability which can cause the application to read past the end of an input stream
  - WithMi uses a Trie structure in decompressing certain compressed files
  - After reading past the end of the input stream WithMi records that a leaf node was created without creating the leaf node
  - In a later stage of decompression, the application continuously searches for the non-existent leaf node
  - This causes the application to exceed the resource usage limit

Additional Detail: AC Vulnerabilities

- **Run Length Encoding (RLE) Negative Integer:** AC Space
  - A variant of the WithMi Application uses a (de)compression method which relies on run length encoding
  - The RLE method compresses consecutive identical bits within a file
    - e.g. “0000” -> “0-4”
    - “0” is the bit value and “4” is the repeat value
  - Decompression Method:

```
decompressedBits = “”;
currentBit = “0”;
repeatValue = 4;
while(repeatValue != 0){
      decompressedBits+=currentBit;
      repeatValue--;
```

  - The decompression method accepts negative repeat values as a valid input
  - Provided a negative repeat value, the decompress method will continuously decrement the value without ever reaching the exit condition of the loop

Additional Detail: AC Vulnerabilities

- **Run Length Encoding (RLE) Integer Overflow:** AC Space
  - A variant of the WithMi Application uses a (de)compression method which relies on run length encoding
  - The RLE method compresses consecutive identical bits within a file
    - e.g. “0000” -> “0-4”
    - “0” is the bit value and “4” is the repeat value
  - The maximum number of bits is a large number stored as a long
  - The number of bits currently decompressed is stored as an integer
  - An adversary can craft an input file which causes the number of bits decompressed variable to exceed the maximum value of an integer (MAX_INT)
  - This will cause the value of the number of decompressed bytes to overflow from MAX_INT to MIN_INT

Additional Detail: Space Vulnerabilities

- SC Space File Transfer
  - When the application is started, users must specify where to look for files to send
  - Files to be sent are split into fixed size chunks, compressed and sent
  - Results in a unique size fingerprint for each of the files to be sent
- SC Space Number of Users
  - When a user is invited to chat they receive a chat state message.
  - Chat state messages are larger than other WithMi messages and their sizes differ based on the number of users in the chat.
  - Results in a unique mapping of message size to number of users in a chat