



Cracking the "Off The Grid" Password Solution

DR. JOSHUA STROSCHEIN

STEPHANIE SLAYDEN

JOSHUA.STROSCHEIN@DSU.EDU

SSLAYDE@KSU.EDU

Presentation Overview



How OTG works



Attacking the Grid



Using CUDA



Testing Methodology



Conclusions/Future Work





The Off The Grid System

Off The Grid

- ▶ Developed by a Security Researcher
- ▶ Used to store passwords off line
- ▶ Uses a randomly generated Latin Square

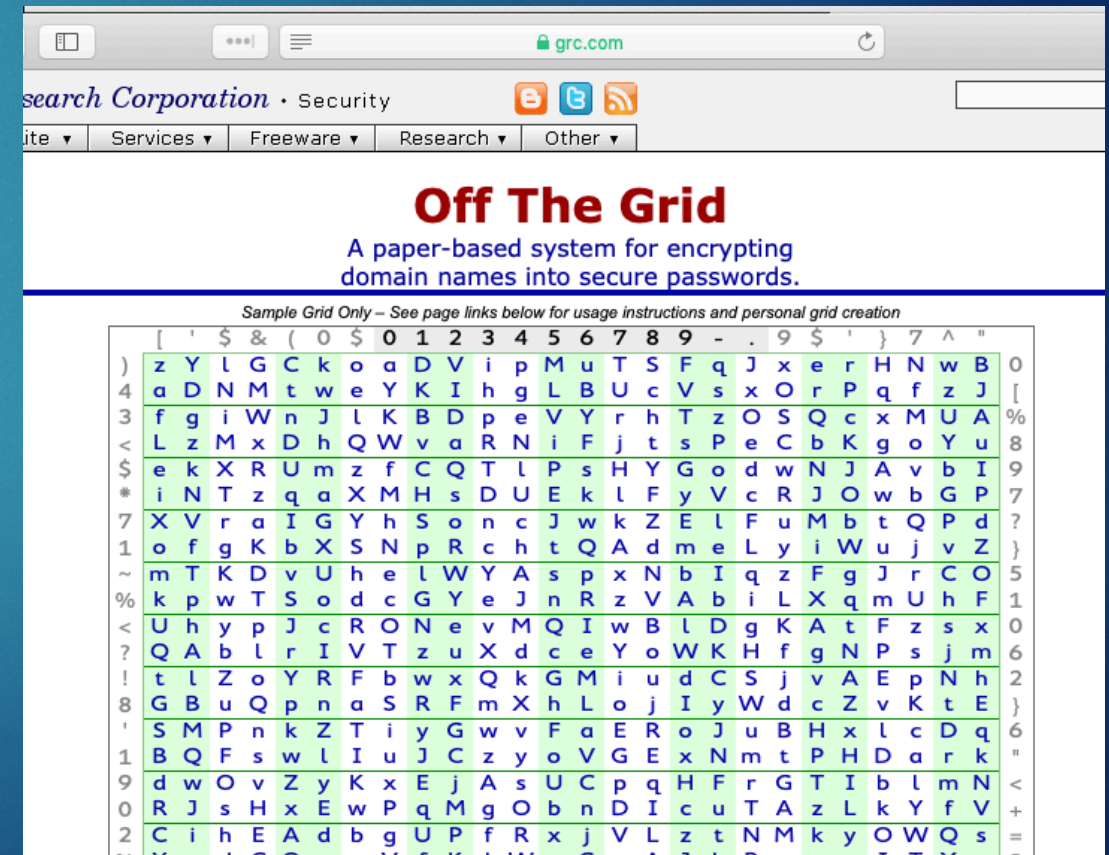
Latin Square

- ▶ An $N \times N$ grid
- ▶ Values 1-N
- ▶ Every row and column
 - ▶ Has N unique Values
 - ▶ No Repeats
- ▶ Similar to Soduku Puzzles
 - ▶ Less restrictive

A	B	D	C
B	C	A	D
C	D	B	A
D	A	C	B

OTG Latin Squares

- ▶ Website allows users to create and print a grid for offline usage
- ▶ <https://www.grc.com/offthegrid.htm>
- ▶ Uses random number generators to create the grid



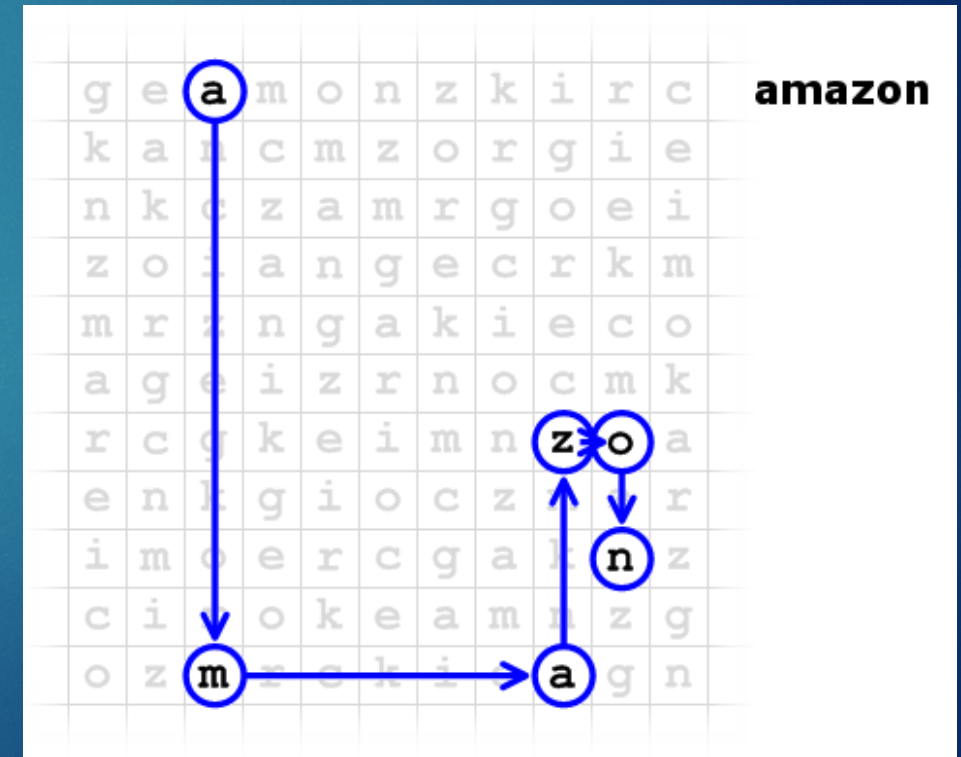
The screenshot shows a web browser window with the URL [grc.com](https://www.grc.com). The page title is "search Corporation • Security". Below the navigation bar, the heading "Off The Grid" is displayed in large red font, followed by the subtitle "A paper-based system for encrypting domain names into secure passwords." Below this, a note reads "Sample Grid Only – See page links below for usage instructions and personal grid creation". The main content is a large grid of characters. The grid has 10 columns and 20 rows. The first row contains the following characters: [' \$ & (0 \$ 0 1 2 3 4 5 6 7 8 9 - . 9 \$ ' } 7 ^ " 0. The subsequent rows contain a mix of uppercase and lowercase letters, digits, and special characters, arranged in a way that suggests a Latin square or a similar cryptographic structure. The grid is used for generating secure passwords by selecting characters from the grid based on a domain name and a key.

Using the OTG System

- ▶ Each domain generates a unique password
- ▶ Use domain to trace through the grid
- ▶ Use the domain to then synthesize the password for that domain
- ▶ Every Domain name will generate a different unique password

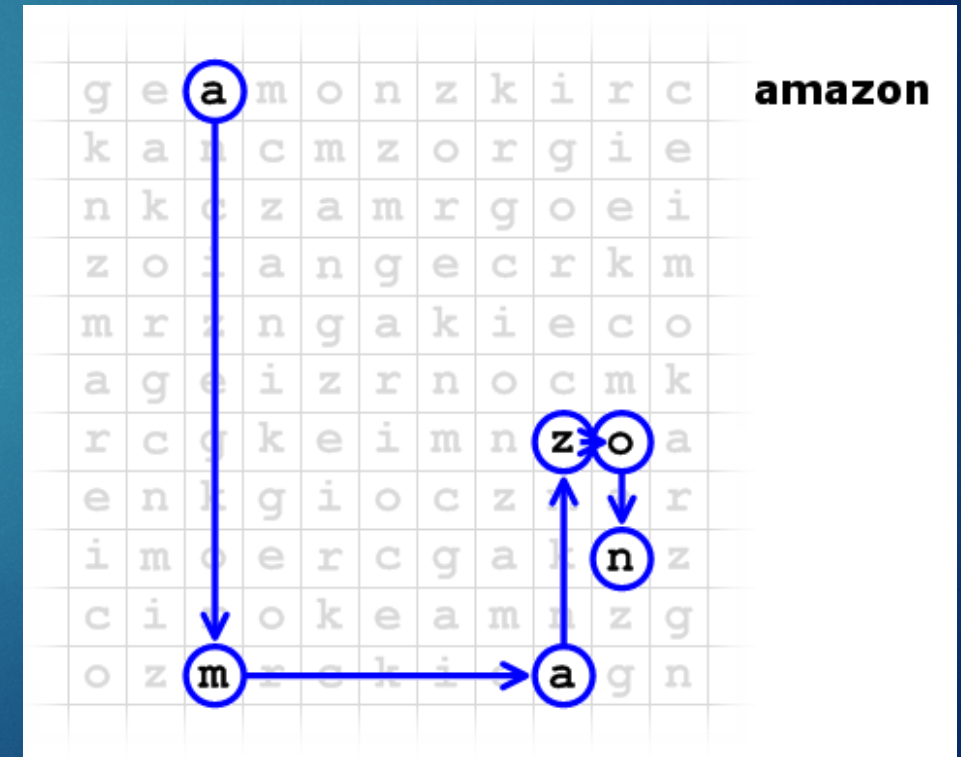
Using the OTG System Phase 1 (tracing)

- ▶ Example
- ▶ Amazon.com
- ▶ Use the letters in AMAZON
 - ▶ Trace through the grid



Using the OTG System Phase 1 (tracing)

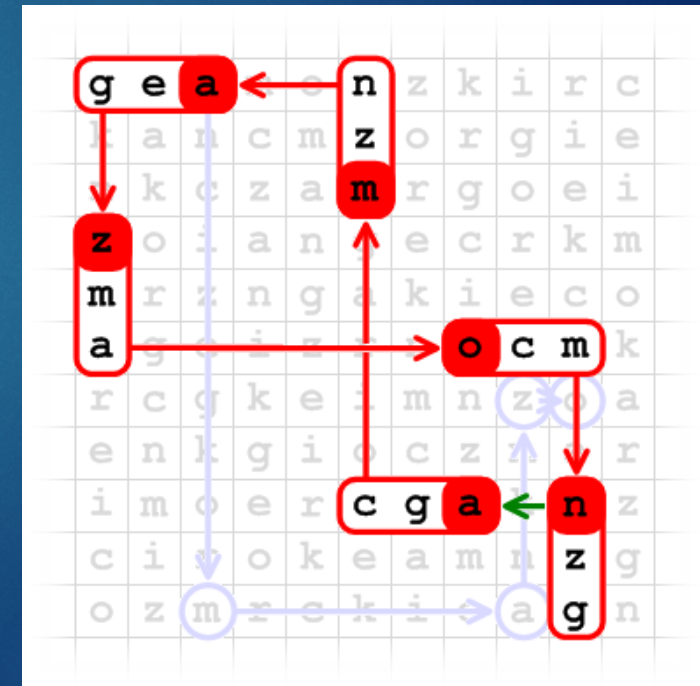
- ▶ Start with A
- ▶ Find the next letter in the domain M
- ▶ Switch from going vertical to horizontal
- ▶ Find the next letter in Domain A
- ▶ Switch from going horizontal to vertical
- ▶ Find the next letter in Domain Z
- ▶ Switch from going vertical to horizontal
- ▶ Repeat until finished



Using the OTG System Phase 2 (synthesizing)

- ▶ Start from the ending point of Tracing (N for AMAZON)
- ▶ Find the first letter in the domain (A)
- ▶ First 2 characters password are generated
 - ▶ The current Direction we moved
 - ▶ GC

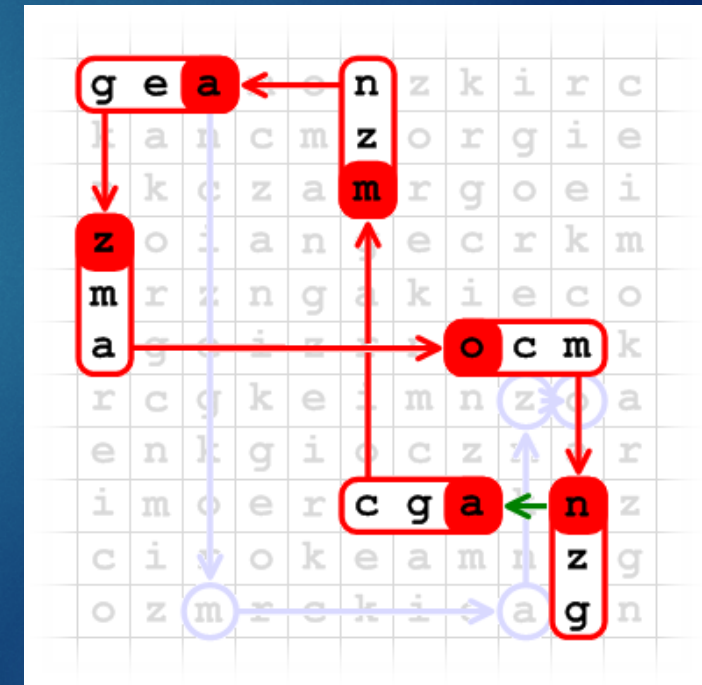
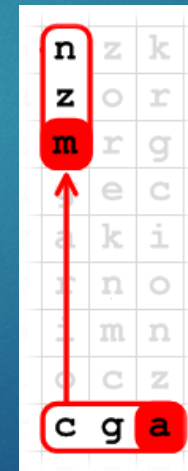
Password for AMAZON.COM
GC



Using the OTG System Phase 2 (synthesizing)

- ▶ Start from the ending point of Tracing (N for AMAZON)
- ▶ Find the first letter in the domain (A)
- ▶ 2 characters password are generated
 - ▶ The current Direction we moved (Called overshoot)
 - ▶ GC
- ▶ Switch Direction
 - ▶ Start with the last character generated
 - ▶ Find the next letter in Domain Name (M)
 - ▶ Next 2 characters of the password
 - ▶ NZ

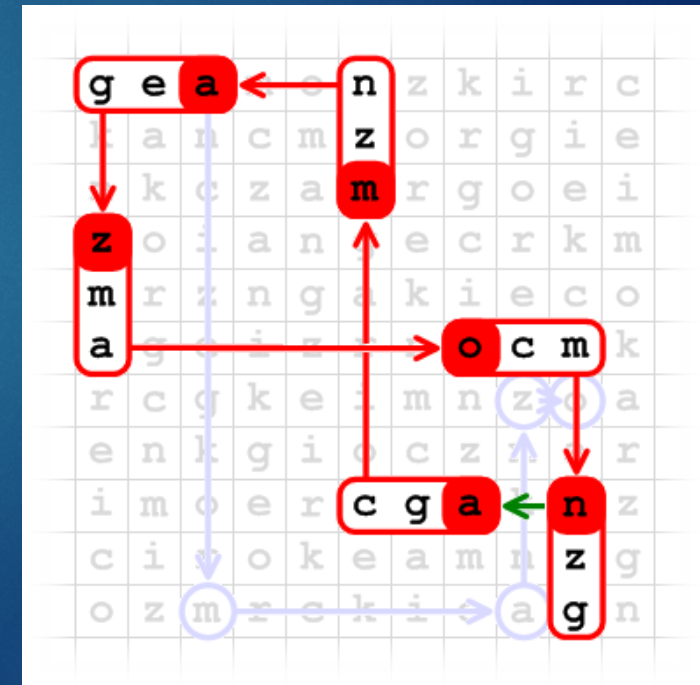
Password for AMAZON.COM
GC ZN



Using the OTG System Phase 2 (synthesizing)

- ▶ Repeat until the last character of the domain is generated
- ▶ A-> EG
- ▶ Z-> MA
- ▶ O-> CM
- ▶ N->ZG

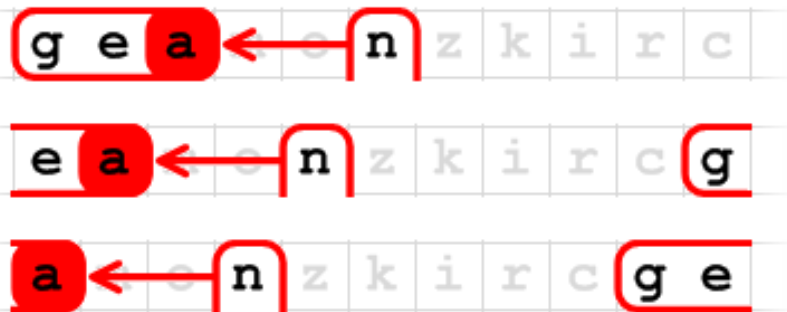
Password for AMAZON.COM
GC ZN EG MA CM ZG



Overshoot Details

- ▶ Overshoot is allowed to wrap around the grid
- ▶ Allows password to be located anywhere

Most of the time, the two characters past our target will be right next to it. But... When the target character is too close to an edge, the two output characters will be found by "wrapping around" to the other side of the grid.



When movement takes us off one edge, we return to the opposite side of the same row or column.

Security Claims

- ▶ Number of possible Latin Squares is tremendous

$$\frac{N!^{2N}}{N^{N^2}}$$

- ▶ For $N = 26$
 - ▶ 9.337×10^{426}
 - ▶ $\log_2 (9.337 \times 10^{426}) \approx 1418$ bits of entropy
- ▶ For OTG Grids
 - ▶ $1.5123 \times 10^{79} \approx 263$ bits of entropy



Attacking the Grid

Attacking the Grid

- ▶ Synthesizing method creates restrictions
- ▶ Restricts the possible grids that could create it
- ▶ Password will be 2x the length of the domain name

Restrictions

- ▶ In the domain Name
- ▶ The last letter N is followed by the consecutive letters AGC
- ▶ $N \rightarrow AGC$
 - ▶ Where the arrow represents 0 or more characters



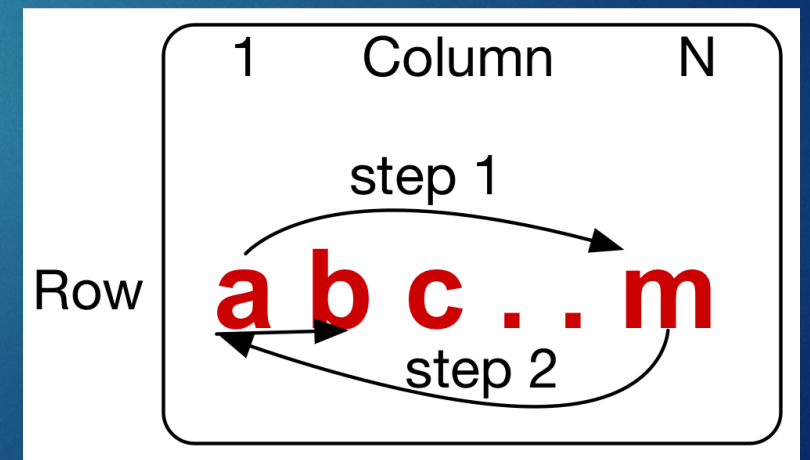
Restrictions

- ▶ In the domain Name
- ▶ The last letter N is followed by the consecutive letters AGC
- ▶ $N \rightarrow AGC$
 - ▶ Where the arrow represents 0 or more characters
- ▶ For amazon and password GC ZN EG MA CM ZG
 - ▶ $N \rightarrow AGC$
 - ▶ $C \rightarrow MZN$
 - ▶ $N \rightarrow AEG$
 - ▶ $G \rightarrow ZMA$
 - ▶ $A \rightarrow OCM$
 - ▶ $M \rightarrow NZG$
- ▶ This represents a chain of restrictions for a single domain



Special Cases

- ▶ Wrapping Around on overshoot
- ▶ What happens when part of the domain is in the password?
 - ▶ The locations are on a border wrap condition
 - ▶ $A \rightarrow MAB$
 - ▶ Creates 2-4 possible locations
- ▶ This example
 - ▶ A is either at row
 - ▶ 1
 - ▶ N
- ▶ Another Example
 - ▶ $A \rightarrow MBA$
 - ▶ 4 possible locations for A
 - ▶ 1, 2, N-1, N





Using CUDA

Using CUDA

- ▶ CUDA is a programming language for NVIDIA graphics cards
- ▶ Allows use to leverage the massive amounts of cores
 - ▶ 1000-4000 per GPU
- ▶ Interfaces between the CPU/RAM and the GPU
- ▶ Generated grids for a known size of possible Latin Squares
 - ▶ Latin Squares of size 10
 - ▶ 34,817,397,894,749,939 possible squares

CUDA Data Structures (Path)

- ▶ Link to the next path
- ▶ Reference to direction
- ▶ Includes the restriction
 - ▶ A→BCD (letters)

```
typedef struct Path
{
    struct Path * next;
    char direction;
    int letters[4];
    char * domain;
    char * pass;
} Path;
```


CUDA Data Structures (Grid)

- ▶ Row
 - ▶ Row restriction
- ▶ Col
 - ▶ Column restriction
- ▶ Array of Cells
 - ▶ Either sentinel value '—'
 - ▶ Value at a location

```
typedef struct Grid
{
    int *row;
    int *col;
    char ** Cells;
    char ok;
} Grid;
```


CUDA Data Structures (Location)

- ▶ X,Y
 - ▶ Current location
- ▶ nextX,nextY
 - ▶ Where to go
- ▶ lastX,lastY
 - ▶ Where the trace ended

```
typedef struct Location
{
    uint8_t x;
    uint8_t y;
    uint8_t nextX;
    uint8_t nextY;
    uint8_t lastX;
    uint8_t lastY;
    uint8_t type;
    uint8_t edge;
} Location;
```


CUDA Data Structures (State)

- ▶ Current state of the System
- ▶ Count and iterations for statistics

```
typedef struct State{  
    Grid grid;  
    Location location;  
    Path * path;  
    long count;  
    long iterations;  
} State;
```


CUDA Program

- ▶ Assume that passwords for sites are leaked
- ▶ Use known Domain Name Password (DNP) pairs
- ▶ Try all possible grids
 - ▶ Distribute work across all CUDA Cores



Methodology and Results

Methodology

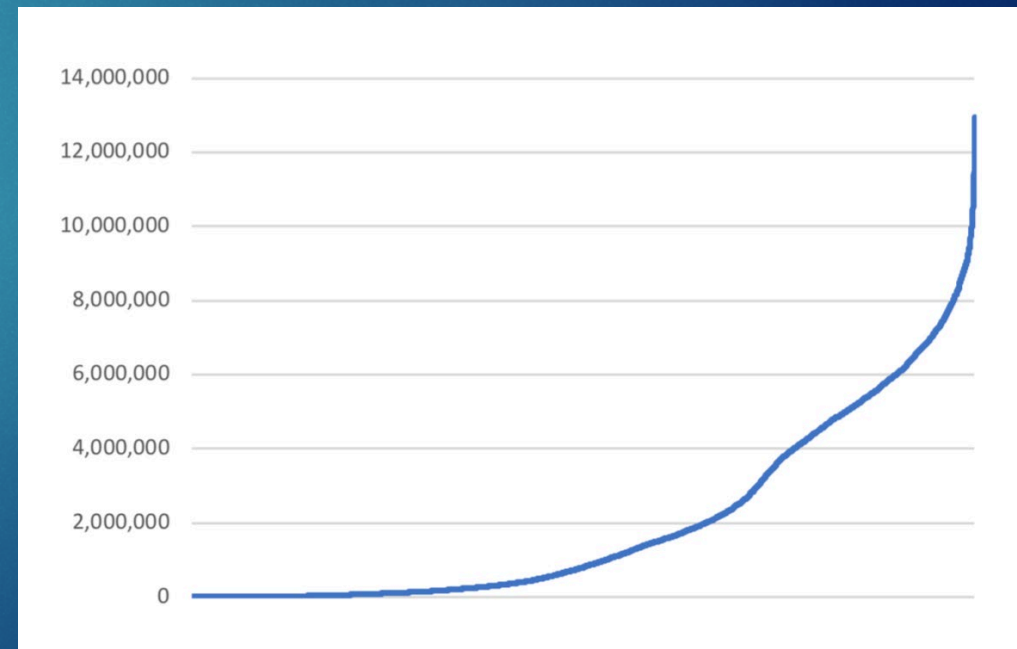
- ▶ Hypothesis
 - ▶ OTG Leaks information with every password leaked
- ▶ Test
 - ▶ Generate a grid for a Latin Square of a known size
- ▶ Determine if information is leaked

Test Square

- ▶ Randomly generated 10 x 10 grid
- ▶ Generate a number of random domain names
 - ▶ Generate the passwords for those domain names
- ▶ Use the OTG cracking software to determine if information is leaked

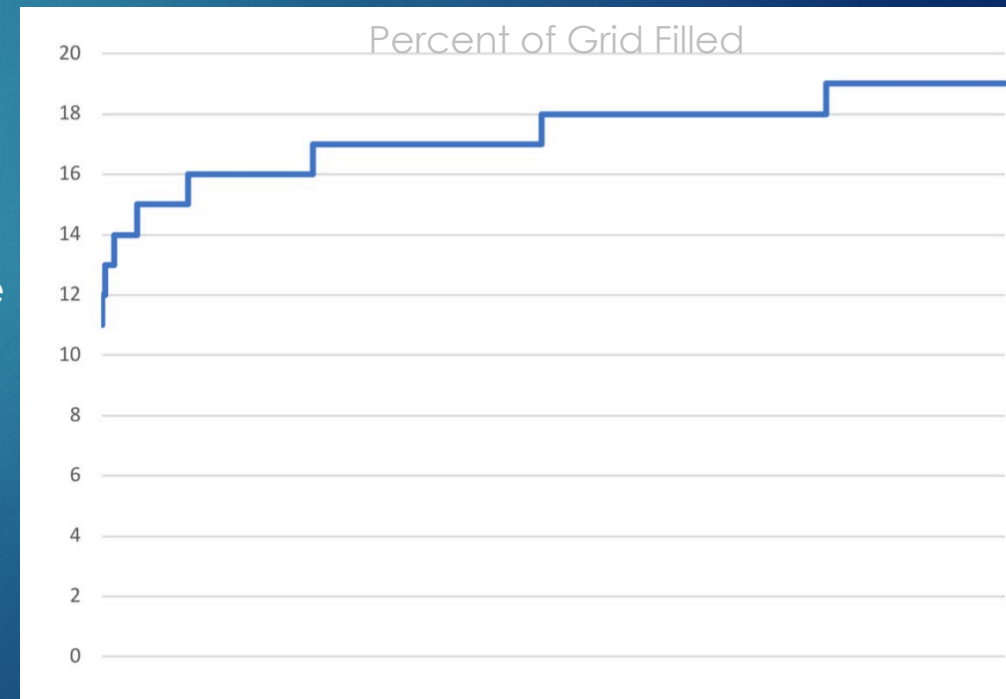
1 Password Possible Grids

- ▶ How many possible grids are there for the 12,288 DNP pairs
- ▶ Range from
 - ▶ 8
 - ▶ Edge cases
 - ▶ 2 possible grids
 - ▶ 4 rotations
 - ▶ 13 million



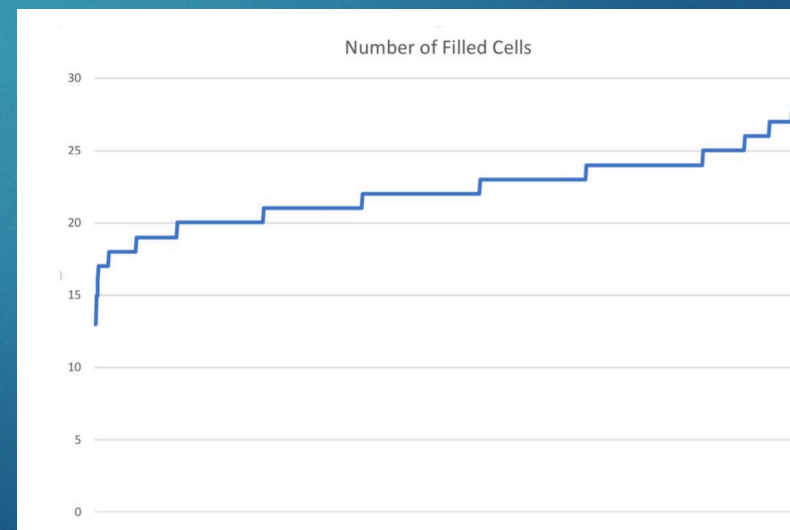
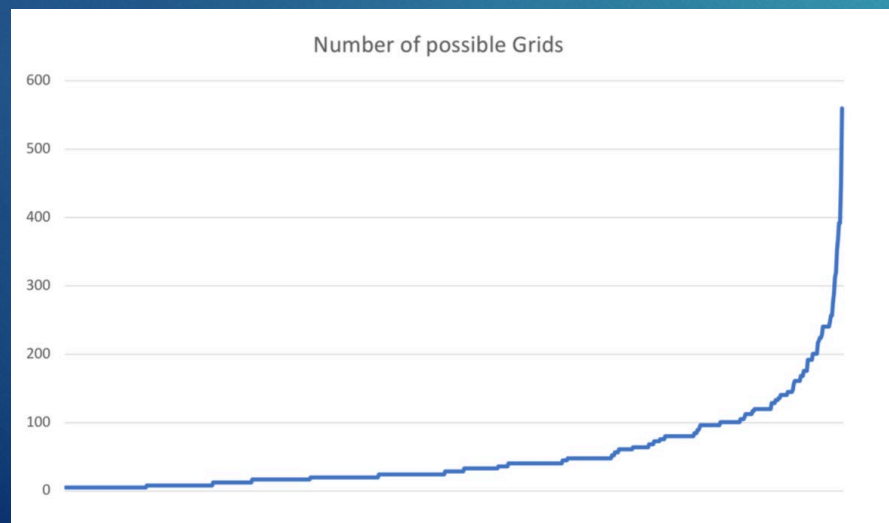
1 Password Percent of Grid Filled

- ▶ Percentage of the grid that is filled in the 12,288 DNP pairs
- ▶ Range from
 - ▶ 11% – 19%
- ▶ Averaged
 - ▶ 16 characters of leakage
- ▶ 51% of grids leaked 18+ characters
 - ▶ 3 characters per character in the domain name



Combining 2 passwords

- ▶ The smallest 40 DNP pairs were combined
 - ▶ These are the worst of the passwords
- ▶ Combined each pair
 - ▶ From 4-560 possible grids
 - ▶ Leaked between 13 and 28 characters from the grid
 - ▶ 2.3 characters of leaked info per character of password



Selecting Optimal Grids

- ▶ Selected grids that had no overlap
 - ▶ Written in Python
- ▶ Combine those grids
- ▶ Yielded better results
 - ▶ Increased to 3.166 characters of leaked info per character of password
- ▶ Took 1.5 days
 - ▶ Single NVIDIA 970

3 domains

- ▶ Using three generated domains
 - ▶ Completed 57 cells
 - ▶ Latin square $\frac{1}{2}$ full can be solved
 - ▶ 2 possible solutions

Combining Grids #4323, #5601 and #5939

-	e	a	g	i	f	b	d	-	c
c	g	h	a	j	e	i	b	d	f
f	j	d	e	h	c	g	i	b	a
e	a	g	h	f	j	d	c	i	b
g	d	j	i	c	b	a	f	e	h
b	f	i	c	g	d	j	h	a	e
d	c	e	b	a	g	h	j	f	i
a	i	c	j	b	h	f	e	g	d
-	b	f	d	e	i	c	a	-	g
i	h	b	f	d	a	e	g	c	j

OTG vs possible Latin Squares

- ▶ Latin squares of size 10
 - ▶ Possible 9.9×10^{36} Latin Squares
- ▶ We were able to
 - ▶ Compute 25% of the grid using 6.6×10^5 computations
 - ▶ Compute 50% of the grid using 1.2×10^6 computations
 - ▶ Solve the rest of the grid
- ▶ Result is 30 orders of magnitude less
 - ▶ 9.9×10^{36} vs 1.2×10^6
- ▶ It would take approximately 19 DNP pairs to crack a OTG System
 - ▶ $26 \times 26 = 676$ cells
 - ▶ $(676 / 2 \text{ (fill 50\% of grid)}) / 3 \text{ (3 characters per domain character)} / 6 \text{ (6 character password)}$



Conclusions

Conclusions

- ▶ This work shows that
 - ▶ The OTG System Leaks information
 - ▶ A system can be designed to take advantage of the flaws in the OTG system
 - ▶ Graphics cards can be used to speed up the process
- ▶ Additional shortfalls of OTG
 - ▶ No multiple logins
 - ▶ No password reset



Future Work

Future Work

- ▶ Increase the size of the grids from 10 to 15
- ▶ Integrate additional tools
 - ▶ Select smallest grids
 - ▶ Snapshot processing
 - ▶ Compile for better hardware
 - ▶ NVIDIA 2080 and beyond



Questions?

Thank You!

DR. MATTHEW MILLER

MILLERMJ@UNK.EDU

DR. JOSHUA STROSCHEIN

JOSHUA.STROSCHEIN@DSU.EDU

STEPHANIE SLAYDEN

SSLAYDE@KSU.EDU