
CTF Code

Writeups

Криптография

24 октября 2021 г.

Оглавление

Easy	1
1 Based task	1
2 Hashes among us	2
Medium	4
1 Please be careful with ASR	4
2 Please don't share	7
Hard	10
1 Do you want to play some gamel?	10
2 Curve task	13

Easy

1 Based task

Теги: baseN, googling

Друг прислал мне какое-то сообщение. Но я не знаю китайского, как и он! Чем же это может быть?

Достаточно простая задача. Нам дается файл с какими-то иероглифами:

𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿𐁀𐁁𐁂𐁃𐁄𐁅𐁆𐁇𐁈𐁉𐁊𐁋𐁌𐁍𐁎𐁏𐁐𐁑𐁒𐁓𐁔𐁕𐁖𐁗𐁘𐁙𐁚𐁛𐁜𐁝𐁞𐁟𐁠𐁡𐁢𐁣𐁤𐁥𐁦𐁧𐁨𐁩𐁪𐁫𐁬𐁭𐁮𐁯𐁰𐁱𐁲𐁳𐁴𐁵𐁶𐁷𐁸𐁹𐁺𐁻𐁼𐁽𐁾𐁿𐂀𐂁𐂂𐂃𐂄𐂅𐂆𐂇𐂈𐂉𐂊𐂋𐂌𐂍𐂎𐂏𐂐𐂑𐂒𐂓𐂔𐂕𐂖𐂗𐂘𐂙𐂚𐂛𐂜𐂝𐂞𐂟𐂠𐂡𐂢𐂣𐂤𐂥𐂦𐂧𐂨𐂩𐂪𐂫𐂬𐂭𐂮𐂯𐂰𐂱𐂲𐂳𐂴𐂵𐂶𐂷𐂸𐂹𐂺𐂻𐂼𐂽𐂾𐂿𐃀𐃁𐃂𐃃𐃄𐃅𐃆𐃇𐃈𐃉𐃊𐃋𐃌𐃍𐃎𐃏𐃐𐃑𐃒𐃓𐃔𐃕𐃖𐃗𐃘𐃙𐃚𐃛𐃜𐃝𐃞𐃟𐃠𐃡𐃢𐃣𐃤𐃥𐃦𐃧𐃨𐃩𐃪𐃫𐃬𐃭𐃮𐃯𐃰𐃱𐃲𐃳𐃴𐃵𐃶𐃷𐃸𐃹𐃺𐃻𐃼𐃽𐃾𐃿𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈𐮉𐮊𐮋𐮌𐮍𐮎𐮏𐮐𐮑𐮒𐮓𐮔𐮕𐮖𐮗𐮘𐮙𐮚𐮛𐮜𐮝𐮞𐮟𐮠𐮡𐮢𐮣𐮤𐮥𐮦𐮧𐮨𐮩𐮪𐮫𐮬𐮭𐮮𐮯𐮰𐮱𐮲𐮳𐮴𐮵𐮶𐮷𐮸𐮹𐮺𐮻𐮼𐮽𐮾𐮿𐯀𐯁𐯂𐯃𐯄𐯅𐯆𐯇𐯈𐯉𐯊𐯋𐯌𐯍𐯎𐯏𐯐𐯑𐯒𐯓𐯔𐯕

- Написать питоновский скрипт

Чтобы райтап был полным, рассмотрим второй вариант, потому что первый достаточно очевидный и не требует пояснений. Скрипт для расшифровки выглядит примерно следующим образом:

Листинг 1: Дешифровка флага

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import base65536
import pybase100 as base100
import base64

def decode(flag):
    flag = base65536.decode(flag)
    flag = base100.decode(flag)
    flag = base64.b64decode(flag)
    flag = flag.decode('utf-8')

    return flag

def main():
    with open("flag.enc", 'r') as flag_file:
        flag = flag_file.read()

    print(decode(flag))

if __name__ == '__main__':
    main()
```

На выходе получаем флаг `oren_ctf_KevinDavidMitnick!`.

2 Hashes among us

Теги: hash crack

Мне тут попалась какая-то странная строка... Говорят, в ней есть флаг. Но как его получить?

Судя по виду зашифрованного флага и названию задачи перед нами какой-то хэш.



```
1775fe677ade895006ac191fc4391761
```

Рис. 4: Слишком много хексов

Если посмотреть длину, то мы увидим, что длина флага ровно 32 символа, что позволяет подумать про MD5. Далее можно либо брутить локально с помощью HashCat/JhonTheRipper или воспользоваться онлайн-сервисами наподобие [CrackStation](#). После чего получаем строку `RobertMorris`, которую нужно обернуть в `oren_ctf_!`. После чего получаем флаг `oren_ctf_RobertMorris!`.

Medium

1 Please be careful with ASR

Теги: RSA, Hastad Attack

Мы с друзьями часто обмениваемся информацией, которую не должен знать никто, кроме нас. Не можете ли вы проверить, насколько хорошо мы справились с этой задачей?

Судя по ключам и названию задачи речь явно идет про RSA. Что же, можно погуглить про атаки на этот алгоритм и наткнуться на весьма интересную штуку под названием **Håstad's broadcast attack**. После чего, если попробовать сдмпить открытую экспоненту и модуль из наших открытых ключей, то можно увидеть, что во всех трех ключах экспонента маленькая и равна 3. Что уже точно намекает на эту атаку.

Суть атаки

Пользователь отправляет зашифрованное сообщение m нескольким пользователям. в данном случае, трём (по числу файлов): $P1, P2, P3$. У каждого пользователя есть свой ключ, представляемый парой «модуль-открытая экспонента» (n_i, e_i) , причём $M < n_1, n_2, n_3$. Для каждого из трех пользователей зашифровывает сообщение на соответствующем открытом ключе и отправляет результат адресату. Атакующий же реализует перехват сообщений и собирает переданные шифртексты (обозначим их как C_1, C_2, C_3), с целью восстановить исходное сообщение M . Значит, по имеющимся трем шифртекстам нужно восстановить сообщение, которое будет флагом.

Почему точно сможем ее реализовать?

Как известно, шифрование сообщения по схеме RSA происходит следующим образом: $C = M^e \pmod{n}$. В случае с открытой экспонентой, равной 3, получение шифртекстов выглядит так:

$$\begin{aligned}C_1 &= M^3 \pmod{n_1} \\C_2 &= M^3 \pmod{n_2} \\C_3 &= M^3 \pmod{n_3}\end{aligned}$$

Зная, что n_1, n_2, n_3 взаимно просты, можем применить к шифртекстам **китайскую теорему об остатках**. Получим в итоге некоторое C' , корень кубический из которого и даст нам искомое сообщение M .

$$C' = M^3 \pmod{n_1 * n_2 * n_3}$$

Вспоминаем, что M меньше каждого из трёх модулей n_i , значит, справедливо равенство:

$$C' = M^3$$

Так мы и найдем наше искомое сообщение M .

Программная реализация атаки Хастада

После всего вышесказанного не составляет труда написать простенький скрипт на питоне, который выдаст зашифрованное сообщение:

Листинг 2: Атака Хастада

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import gmpy2
gmpy2.get_context().precision = 2048

from binascii import unhexlify
from functools import reduce
from gmpy2 import root
from Crypto.PublicKey import RSA

def chinese_remainder_theorem(items):
    N = 1
    for a, n in items:
        N *= n

    result = 0
    for a, n in items:
        m = N // n
        r, s, d = extended_gcd(n, m)
        if d != 1:
            raise "Input_not_pairwise_co-prime"
        result += a * s * m

    return result % N

def extended_gcd(a, b):
    x, y = 0, 1
    lastx, lasty = 1, 0

    while b:
        a, (q, b) = b, divmod(a, b)
```

Medium

```
x, lastx = lastx - q * x, x
y, lasty = lasty - q * y, y

return (lastx, lasty, a)

def mul_inv(a, b):
    b0 = b
    x0, x1 = 0, 1
    if b == 1:
        return 1
    while a > 1:
        q = a // b
        a, b = b, a % b
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += b0

    return x1

def get_cipher(filename):
    with open(filename, 'rb') as cipher:
        value = cipher.read().hex()

    return int(value, 16)

def get_modulus(filename):
    with open(filename) as keyfile:
        keystr = keyfile.read()
        key = RSA.import_key(keystr)

    return key.n

def main():
    ciphertext1 = get_cipher("flag.enc.alice")
    ciphertext2 = get_cipher("flag.enc.bob")
    ciphertext3 = get_cipher("flag.enc.eve")

    modulus1 = get_modulus("alice.pub")
    modulus2 = get_modulus("bob.pub")
    modulus3 = get_modulus("eve.pub")
```


Medium

```
C = chinese_remainder_theorem([(ciphertext1, modulus1),
                                (ciphertext2, modulus2),
                                (ciphertext3, modulus3)])

flag = int(root(C, 3))
flag = hex(flag)[2:]

print(unhexlify(flag).decode('utf-8'), end='')

if __name__ == '__main__':
    main()
```

После выполнения скрипта на выходе получаем расшифрованный текст с флагом:

CIH, also known as Chernobyl or Spacefiller, is a Microsoft Windows 9x computer virus which first emerged in 1998. Its payload is overwriting critical information on infected system drives, and in some cases destroying the system BIOS. Flag: oren_ctf_CIH!

2 Please don't share

Теги: Shamir's Secret Sharing, AES

Этот сервер отдает флаг, но как-то странно. Я никак не могу понять, какое тут шифрование и как вообще его можно взломать...
nc ctf-edu-t.orb.ru 2386

При подключении к серверу мы видим сообщение

```
Here's a base64-encoded and encrypted flag: KSlIe5gsRMTPiAulWr5pDpXFz8WrL9vq0HpDRkokWVhsIxzzWqF6Cfo7tt01br8H
You need a secret and literally zero ivent to get it!

Here is three part:
Part 1: 1-1510d2b3cacd8f387ff5b7eb52900b5ff60241072443620f7ae55d0efccc9fbb
Part 2: 2-42a67d81d71fae91aeb79dd30d61cd12f2afaf3c01d7102bf7df8ce069cad82d
Part 3: 3-88c1006a24f65e0b8c45b1b730754519db7640440f45c4af86ce1e8854b4ccf0
```

Рис. 1: Выглядит странно

Исходя из названия задачи и формулировки подсказки можно предположить, что речь идет о **Shamir Secret Sharing Scheme** (или иногда можно встретить сокращение SSSS). Помимо этого есть опечатка в слове **event**, что тоже дает какую-то подсказку. По подсказке понятно, что схемой разбит не сам флаг, а ключ для какого-то алгоритма шифрования с нулевым вектором инициализации, которым уже зашифрован флаг.

Немного о схеме Шамира

Если очень кратко, то секрет (информация, которой мы хотим поделиться) S разбивается на n частей таким образом, чтобы можно было восстановить исходное сообщение S только в случае, если мы имеем не менее k кусочков из n . Даже наличие $k - 1$ кусков недостаточно, чтобы восстановить исходную S . Чуть более подробно про это написано вот [здесь](#).

Реализация

Можно писать разделение секрета самому, но это выходит за рамки данного райтапа. Если немного погуглить, то можно найти достаточно большое количество реализаций этого алгоритма на самых разных языках. Для питона существует вот [эта](#) реализация, но, к сожалению, она имеет проблемы с третьей версией из-за использования типа `long`, который был удален. Но ничего не мешает написать скрипт на втором. Итак, после получения полного секрета можно заметить, что это строка длиной 16 символов, что, в купе с нулевым инициализирующим вектором, дает возможность предположить, что используется AES-128. Остается лишь понять, какой из двух самых известных вариантов используется - ECB или CBC. Это можно сделать просто попробовав каждый из них. После чего можно написать небольшой питоновский скрипт, чтобы получить флаг:

Листинг 3: Расшифровка флага AES-CBC

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

import sys
from base64 import b64decode
from Crypto.Cipher import AES
from secretsharing import SecretSharer

def main():
    if len(sys.argv) < 4:
        print("Usage: _{_}_part1_part2_part2".format(sys.argv[0]))
        exit(1)

    AES_KEY = SecretSharer.recover_secret([sys.argv[1], sys.argv[2], sys.argv[3]])
    IV = b'\x00' * 16
    cipher = AES.new(key=AES_KEY, mode=AES.MODE_CBC, iv=IV)

    with open("flag.enc", 'r') as flagfile:
        enc_flag = b64decode(flagfile.read())

    flag = cipher.decrypt(enc_flag)

    print("Decrypted_flag_is:_{_}".format(flag))
```

Medium

```
if __name__ == '__main__':  
    main()
```

После выполнения скрипта на выходе получаем расшифрованный флаг:
Decrypted flag is: oren_ctf_ILOVEYOU_or_LoveLetter!

Hard

1 Do you want to play some gamel?

Теги: Elgamal

Справитесь ли вы с Великим Оракулом? Или, хотя бы, убедите его отдать флаг
nc ctf-edu-t.orb.ru 2263

При подключении к серверу мы видим сообщение

```
Hi. This is your friendly "Decryption Oracle"
We have implemented a well-known public-key cryptosystem. Guess which ;)

Modulo: 10249196104423346754012526841431517045053043020097
Generator: 7002822320099715830030518845294150206874169225271
Public key: 288068429935564498080808053287570428720268163407
Ciphertext: (2856765715487887807732676422667691564782704134025, 1504629955116654156493298697094141235221858422324336317235357801581913183478482393057441378)

Insert your Ciphertext-Tuple for me to decrypt - comma seperated (e.g. 5,6)
>>> █
```

Рис. 1: Опять какая-то асимметрия

Судя по тому, что шифротекст разделен на два числа - перед нами криптосистема Эль-Гамала.

Криптосистема Эль-Гамала

Идея данной системы основана на сложности нахождения целого неотрицательно-го числа x , удовлетворяющего уравнению $g^x = a$ в циклической группе¹ (вообще говоря, такая операция называется дискретным логарифмированием).

Генерация ключа

Алгоритм генерации ключа работает по следующему принципу:

¹Тут нужно некоторое уточнение. Циклическая группа $(G, *)$, если не вдаваться в дебри высшей математики, - множество для элементов которого мы определили по каким правилам можно их складывать (строго говоря, то сложение которое проходят в школе и это - немного разные вещи. В данном случае под сложением подразумевается операция, которая однозначно ставит в соответствие двум элементам e_1 и e_2 группы G некоторый элемент e_3 , также принадлежащий группе G). Эта группа особенна тем, что результат операции может быть меньше операндов. По поводу цикличности все совсем просто - группа образуется некоторым элементом g , который называется образующим, а все остальные ее элементы это степени образующего. То есть $G = \{g^1, g^2, \dots\}$. Группа становится циклической, если в $g^n = g$. Тогда число n называется порядком группы.

Hard

- Выбираем случайное большое целое число q и строим циклическую группу G_q с образующим элементом g .
- Выбираем случайный целочисленный положительный x , т. ч. $1 \leq x \leq q - 1$ и $(x, q) = 1$
- Считаем $h = g^x$
- Публичный ключ состоит из трех частей: (q, g, h)

Шифрование

Пусть мы хотим передать некоторое сообщение M . Тогда:

- Выбирается случайный целочисленный положительный y , $1 \leq y \leq q - 1$ и $(y, q) = 1$
- Вычисляется секрет $s = h^y = (g^x)^y = g^{xy}$
- Вычисляется $c_1 = g^y$
- Вычисляется $c_2 = M * s$

Шифротекстом будет пара (c_1, c_2)

Дешифровка

Мы хотим расшифровать сообщение (c_1, c_2) и у нас есть приватный ключ x . Если провести некоторые математические выкладки:

$$\begin{aligned} &\begin{cases} c_1 = g^y \\ c_2 = M * s \end{cases} \\ &\Rightarrow \\ &\begin{cases} c_1 = g^y \\ c_2 = M * h^y \end{cases} \\ &\Rightarrow \\ &\begin{cases} c_1 = g^y \\ c_2 = M * g^{xy} \end{cases} \end{aligned}$$

То есть для получения оригинального сообщения M необходимо произвести следующие операции:

$$M = \frac{c_2}{c_1^y} = \frac{M * g^{xy}}{g^{xy}}.$$

Атака

Очевидно, что тот же самый шифротекст система отказывается расшифровывать. Но можно немного схитрить - умножить и поделить дробь на два. От этого, как известно, результат не поменяется:

Hard

$$M * 2 = \frac{c'_2}{c'_1} = \frac{c_2 * 2}{c_1} = \frac{M * g^{xy} * 2}{g^{xy}}.$$

После чего останется только поделить сообщение на два и получить флаг!

Реализация

Исходя из всего вышеперечисленного, написать простенький скрипт на питоне не составляет труда. Для упрощения самому себе жизни я буду использовать библиотеку **pwntools** (вообще говоря, она задумывалась для работы с бинарщиной, но также через нее весьма удобно работать с сокетами):

Листинг 4: Взлом криптосистемы Эль-Гамала

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
import re

def get_cipher(msg):
    str_tp = re.search(r'\d*,_\d*', msg).group(0)

    return tuple(map(int, str_tp.split(',')))

def mul_tuple(tuple):
    new_tuple = (tuple[0], tuple[1] * 2)

    return ',_'.join(map(str, new_tuple))

def get_decrypt_flag(enc_flag):
    hexflag2 = ''.join('{:02x}'.format(ord(ch)) for ch in enc_flag)
    numflag2 = int(hexflag2, 16)
    numflag = numflag2 // 2
    hexflag = hex(numflag)[2:]

    return ''.join([chr(int(''.join(ch), 16)) \
                    for ch in zip(hexflag[0::2], hexflag[1::2])])

def main():
    r = remote('localhost', 1488)

    msg = r.recvuntil('>>>_').decode('utf-8')
```

```

tuple = get_cipher(msg)

msg_for_flag = mul_tuple(tuple)
r.sendline(msg_for_flag)

enc_flag = r.recvline().decode('utf-8')
flag = get_decrypt_flag(enc_flag)

print("Decrypted_flag : {}".format(flag))

r.close()

if __name__ == '__main__':
    main()

```

На выходе скрипта получаем флага `oren_ctf_WannaCry!`

2 Curve task

Теги: Diffie-Hellman, Elliptic curve

Мы перехватили трафик общения двух хакеров, которые явно что-то замышляют. Одного из них удалось взять горячим, даже получилось вытащить из его системы клиент, через который они общались. Но вот как прочитать их переписку... Задержанный не хочет идти на контакт со следствием и только ехидно говорит, что мы никогда не взломаем их сверх-продвинутый крипто-алгоритм. Помогите!

В задаче применяется алгоритм Диффи-Хеллмана, но немного усложненная версия на эллиптических кривых. Традиционная аббревиатура: ECDH (Elliptic Curve Diffie-Hellman). Подробнее про криптографию на эллиптических кривых можно почитать [вот тут](#). Нам дается клиент и сетевой дамп его общения с сервером.

Алгоритм клиента выглядит следующим образом:

- Подключение к серверу и получение параметра кривой: a, b, p (уравнение кривой выглядит как $y^2 = x^3 + a * x + b \pmod{p}$).
- Получение начальной точки G
- Подтверждение начальной точки (или отказ от нее и предложение своей)
- Вычисление секретного ключа a как случайного числа
- Отправка $G * a$

Hard

- Получение от сервера $G * d$ (где d - секретный ключ сервера)
- Вычисление общего ключа $s = a * (dG) = adG$

Сессия выглядит следующим образом:

```
Establishing secure connection via ECDH...
32593,1864507527,560147175803221327
2309607047,53492280
Accept? [y/n]
y
80842207261242668,482348047571127292
411081575920557992,302474624156986014
0RIGdbNapocC9vboA6JibpPFofB0111RQA10aSzSg5zxEjA8K2QFJpBDZTWOWQ==
jxJ263lpDOs1iu8PsFHFryjNxc7LwKeXuSwlx2/Sjw==
cDxuXhw=
ugI=
gdOXQjD8ruh6VRTj0topdqS3SAUKy+1nee2XFCpZA/s=
uVf4eg==
uAVKR1ir
b6dQ3YQ0RWHPN0Z7eRKNDLLCMoF59iJM/cwxaMyADGfxYjHqrQenQ==
njdw
SrmN
```

Рис. 2: Где-то здесь точно есть флаг

Как можно заметить, модуль p не очень большой: 560147175803221327, это примерно 59 бит. Поэтому в данном случае заходит решение "в лоб" через вычисление дискретного логарифма (см. предыдущую задачу). Можно попробовать самому реализовать какой-то из вариантов, но проще всего использовать `sage`:

```
sage: p = 560147175803221327
sage: E = EllipticCurve(GF(p), [32593, 1864507527])
sage: g = E(2309607047, 53492280)
sage: v1 = E(411081575920557992, 302474624156986014)
sage: v2 = E(80842207261242668, 482348047571127292)
sage: g.discrete_log(v1) * v2
(263085750118593959 : 99484538495976508 : 1)
sage: g.discrete_log(v2) * v1
(263085750118593959 : 99484538495976508 : 1)
sage:
```

Рис. 3: Где-то здесь точно есть флаг

После чего не составляет труда, используя данную библиотеку **cryptor** реализовать процесс дешифровки общения и получить флаг:

Листинг 5: Секреты перестают быть таковыми

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

import cryptor

def decrypt(cipher, crypted):
    return cipher.decrypt(crypted)

def main():
    server_msgs = [
        "0RIGdbNapocC9vboA6JibpPFofBO111RQA10aSzSg5zxEjA8K2QFJpBDZTWOWQ==",
        "jxJ263lpDOs1iu8PsFHFYjNxc7LwKeXuSwlx2/Sjw==",
        "uGI=",
        "uVf4eg==",
        "b6dQ3YQ0RWHPN0Z7eRKNDILCMoF59iJM/cwxaMyADGfxYjHqrQenQ==",
        "SrmN"
    ]
    client_msgs = [
        "cDxuXhw=",
        "gdOXQjD8ruH6VRtj0topdqS3SAUKy+1nee2XFCpZA/s=",
        "uAVKR1ir",
        "njdw"
    ]

    priv_x = 263085750118593959
    priv_y = 99484538495976508

    client_cipher = cryptor.cryptor(priv_x, priv_y, "client")
    server_cipher = cryptor.cryptor(priv_x, priv_y, "server")

    print(decrypt(client_cipher, server_msgs[0]))
    print(decrypt(client_cipher, server_msgs[1]))
    for i in range(4):
        print(decrypt(server_cipher, client_msgs[i]))
        print(decrypt(client_cipher, server_msgs[i + 2]))

if __name__ == "__main__":
```

Hard

```
main()
```

После чего можно увидеть о чем же общались хакеры:

```
Successfully established the secure connection
```

```
Waiting for operator to jump in
```

```
Hello
```

```
Hi
```

```
Can you give me the flag please?
```

```
Sure
```

```
Great!
```

```
Here is your flag: oren_ctf_TuringBombe!
```

```
Ty!
```

```
Bye
```